01807.001774. PATENT APPLICATION

#2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
HERVÉ RUELLAN ET AL.	)	Examiner: N.Y.A.
Application No.: N.Y.A.	)	Group Art Unit: N.Y.A.
Filed: HEREWITH	)	
For: METHOD AND DEVICE FOR PARTITIONING A COMPUTER PROGRAM	)	January 14, 2002

Commissioner for Patents
Washington, D.C. 20231

CLAIM TO PRIORITY
AND
SUBMISSION OF PRIORITY DOCUMENT

Sir:

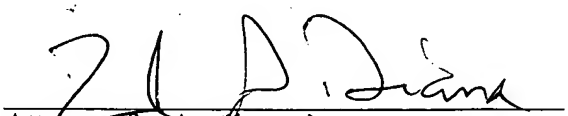
Applicants hereby claim priority under the International Convention and all rights to which they are entitled under 35 U.S.C. § 119 based upon the following French Priority Application:

0100532, filed January 16, 2001.

A certified copy of the priority document is enclosed.

Applicants' undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,



Attorney for Applicants

Registration No. 29,486

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

NY_MAIN 230735 v 1

R E P U B L I Q U E F R A N Ç A I S E

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 04 JAN. 2002

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (1) 53 04 53 04
Télécopie : 33 (1) 42 93 59 30
www.inpi.fr

ETABLISSEMENT PUBLIC NATIONAL CREE PAR LA LOI N° 51-444 DU 19 AVRIL 1951

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ
Code de la propriété intellectuelle - Livre VI

cerfa N° 11354*01

REQUÊTE EN DÉLIVRANCE 1/2

Cet imprimé est à remplir lisiblement à l'encre noire DB 540 W /190600

Réservé à l'INPI

REMISE DES PIÈCES
DATE 16 JAN 2001
LIEU 75 INPI PARIS

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0100532

DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 16 JAN. 2001

Vos références pour ce dossier *(facultatif)* BIF022759/FR

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

RINUY, SANTARELLI
14, avenue de la Grande Armée
75017 PARIS

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	Cochez l'une des 4 cases suivantes
Demande de brevet	☒
Demande de certificat d'utilité	☐
Demande divisionnaire	☐
Demande de brevet initiale	N° Date / /
ou demande de certificat d'utilité initiale	N° Date / /
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N° Date / /

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Procédé et dispositif de partition de programme informatique.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	
	Pays ou organisation Date / / N°
	Pays ou organisation Date / / N°
	Pays ou organisation Date / / N°
	☐ **S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»**

5 DEMANDEUR	☐ **S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»**
Nom ou dénomination sociale	CANON KABUSHIKI KAISHA
Prénoms	
Forme juridique	Société de droit Japonais
N° SIREN	
Code APE-NAF	
Adresse — Rue	30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, JAPON
Adresse — Code postal et ville	
Pays	JAPON
Nationalité	JAPONAISE
N° de téléphone *(facultatif)*	
N° de télécopie *(facultatif)*	
Adresse électronique *(facultatif)*	

Remplir impérativement la 2ème page

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI

REMISE DES PIÈCES
DATE 16 JAN 2001
LIEU 75 INPI PARIS

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0100532

DB 540 W /260899

Vos références pour ce dossier : *(facultatif)*		BIF022759/FR
6 MANDATAIRE		
Nom		
Prénom		
Cabinet ou Société		RINUY, SANTARELLI
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	14 AVENUE DE LA GRANDE ARMEE
	Code postal et ville	75017 PARIS
N° de téléphone *(facultatif)*		01 40 55 43 43
N° de télécopie *(facultatif)*		
Adresse électronique *(facultatif)*		
7 INVENTEUR (S)		
Les inventeurs sont les demandeurs		☐ Oui ☒ Non **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 RAPPORT DE RECHERCHE		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		☒ ☐
Paiement échelonné de la redevance		**Paiement en deux versements, uniquement pour les personnes physiques** ☐ Oui ☐ Non
9 RÉDUCTION DU TAUX DES REDEVANCES		**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence) :*
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire)

Bruno QUANTIN 92.1206
RINUY, SANTARELLI

VISA DE LA PRÉFECTURE OU DE L'INPI

A. TROUDART

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

La présente invention concerne d'une manière générale les procédés et dispositifs de partition de programmes informatiques du type de ceux utilisés pour la conception de programmes informatiques distribués.

Plus particulièrement, la présente invention vise la partition de programmes informatiques susceptibles d'accéder à des informations réparties sur différents sites d'un réseau informatique. De façon classique, ces informations proviennent de sources de données telles que par exemple des fichiers informatiques, des bases de données ou des entrées utilisateur.

Il convient tout d'abord de noter que, dans le cas de tels programmes informatiques, de nombreux paramètres peuvent affecter le temps de transfert et de traitement de ces informations. Parmi ces paramètres, on trouvera, et de manière non exhaustive, des caractéristiques propres aux informations elles-mêmes, tel que le volume des informations à traiter, mais aussi des caractéristiques des réseaux de communication reliant les différents sites, et bien sur des caractéristiques propres aux ordinateurs mettant en œuvre ces programmes. La conception de tels programmes est donc particulièrement complexe.

Les procédés et dispositifs de partition de programmes informatiques classiques apportent une aide à la conception de tels programmes distribués. De façon connue, ils permettent, d'une part, de définir un modèle de réalisation de l'application distribuée, et d'autre part, de mener une analyse, par un certain nombre de mesures, du comportement de l'application dans ce modèle de réalisation.

On connaît, par exemple par le brevet américain US 5,724,556, une méthode de conception de programme distribué accédant à des informations susceptibles d'être réparties sur différents sites. Cette méthode permet à l'utilisateur de définir une partition de ces informations en sous-ensembles d'informations, une partition des instructions du programme distribué en modules et de regrouper ces sous-ensembles d'informations et ces modules en sous-programmes. Le concepteur peut alors définir un modèle de programme distribué correspondant à une répartition particulière de ces sous-programmes sur différents sites.

La méthode permet par la suite l'analyse du modèle ainsi défini, par la production de données représentatives des performances de l'application par exemple. Dans le cas où les résultats de cette analyse ne sont pas jugés satisfaisants, l'utilisateur peut concevoir un nouveau modèle et répéter, sur ce nouveau modèle, la phase d'analyse.

Cette méthode précédemment décrite permet donc d'assister la conception d'applications distribuées, notamment celles accédant à de l'information répartie sur différents sites. Cependant, elle ne permet pas une partition automatique, mais met en œuvre au contraire un processus manuel et itératif nécessitant, à chaque étape, la conception d'un nouveau modèle par l'utilisateur.

La présente invention a pour objet, d'une manière générale, un procédé et un dispositif permettant de résoudre, plus avantageusement, ces problèmes de conception.

Plus précisément, l'invention concerne un procédé de partition d'un programme informatique situé sur un premier site de traitement, le programme comportant des sous-programmes susceptibles de transférer des informations, caractérisé en ce qu'il comporte une étape de détermination automatique, pour au moins l'un desdits sous-programmes, de données représentatives du transfert d'au moins une partie des informations traitées par ledit sous-programme, et une étape d'affectation dudit sous-programme à un deuxième site de traitement en fonction desdites données.

Corrélativement l'invention concerne un dispositif de partition d'un programme informatique situé sur un premier site de traitement, le programme comportant des sous-programmes susceptibles de transférer des informations, caractérisé en ce qu'il comporte des moyens de détermination automatique, pour au moins l'un desdits sous-programmes de données représentatives du transfert d'au moins une partie des informations traitées par ledit sous-programme et des moyens d'affectation dudit sous-programme à un deuxième site de traitement en fonction desdites données.

De façon avantageuse, la présente invention permet une partition automatique du programme informatique. Grâce à la phase de détermination automatique des partitions, l'utilisateur n'a pas besoin de définir des modèles jusqu'à obtenir un modèle dont les résultats sont jugés satisfaisants. Le processus itératif de conception de modèles qui peut être long et fastidieux est ainsi évité. Par exemple, le résultat de la phase de détermination automatique des partitions est une liste de points de coupure du programme informatique permettant de déterminer les sous-programmes à transférer et un site d'affectation pour chacun de ces sous-programmes.

Selon une caractéristique préférée, l'étape de détermination automatique des dites données représentatives comporte une sous-étape de modification du code source du programme informatique, une sous-étape de compilation du code modifié et une étape d'obtention desdites données représentatives par au moins une exécution dudit programme modifié.

La modification du code source, autrement appelée « instrumentation du code source », permet en particulier d'ajouter des variables au programme informatique et de créer une structure de données permettant de mémoriser automatiquement les dites données représentatives du transfert d'informations traitées par les différents sous-programmes. Cette structure de données permet en particulier de mémoriser les transferts d'information entre deux sous-programmes ou entre un sous-programme et une source de données située sur un site distant.

Selon une autre caractéristique préférée, les données représentatives sont obtenues de façon statistique, après au moins deux exécutions dudit programme informatique modifié.

Cela permet en particulier d'obtenir des données représentatives de transferts de différentes quantités d'informations et d'améliorer la fiabilité de l'étape d'affectation.

Selon une autre caractéristique, les données représentatives prennent en compte des caractéristiques d'un canal de transmission entre le premier site de traitement et le deuxième site de traitement, ces caractéristiques étant choisies parmi la latence, la bande passante, le taux d'erreur, la charge moyenne du canal de transmission, et une valeur dépendante d'un protocole de communication.

En faisant varier automatiquement ces différentes caractéristiques pour chaque exécution du programme modifié, le concepteur de programme distribué peut ainsi obtenir une partition qui reste performante même si les conditions du réseau varient.

L'invention concerne aussi un appareil de traitement de programme informatique incluant le dispositif de partition, ou des moyens de mise en œuvre du procédé de partition.

Un moyen de stockage d'information, lisible par un ordinateur ou par un microprocesseur, intégré ou non au dispositif, éventuellement amovible, mémorise un programme mettant en œuvre le procédé de partition.

On comprendra mieux l'invention à la lumière de la description qui va suivre donnée à titre d'exemple et faite en référence aux figures annexées dans lesquelles :

- la figure 1 est un mode de réalisation d'un dispositif mettant en œuvre l'invention ;

- la figure 2 est une représentation détaillée d'un dispositif conforme à l'invention ;

- la figure 3a est un exemple de programme informatique susceptible de subir une partition conforme à l'invention ;

- la figure 3b représente le programme de la figure 3a après partition ;
- la figure 4 est un mode de réalisation du procédé conforme à l'invention ;
- la figure 5a est un exemple de programme sous forme de code source ;
- la figure 5b représente le code source de la figure 5a après instrumentation conforme à l'invention ;
- les figures 6a, 6c et 6d détaillent des fonctions spécifiques substituées, pendant la phase d'instrumentation d'un code source, aux fonctions de bas niveau du tableau 220 de la figure 2 ;
- la figure 6b est un tableau d'information sur la localisation des sources de données ;
- les figures 7a et 7b sont des tableaux regroupant les données représentatives du transfert d'information ;
- la figure 8 est un graphe d'appel obtenu à partir du tableau de la figure 7b ;
- les figures 9a et 9b sont des exemples de graphes d'appel ;
- la figure 10 représente le graphe de la figure 8 après insertion de points de coupure conformes à l'invention;
- la figure 11 représente les étapes d'un algorithme de détermination des points de coupure selon l'invention ; et
- la figure 12 représente les étapes d'un algorithme de détermination d'un site d'affectation selon l'invention.

Selon le mode de réalisation choisi et représenté à la figure 1, un dispositif mettant en œuvre l'invention est par exemple un micro-ordinateur 105.

Le dispositif 105 comporte une interface de communication 112 reliée à un réseau 113 apte à transmettre des programmes informatiques destinés à subir une partition ou inversement à transmettre des résultats d'une telle partition sous la forme de fichiers contenant des points de coupure.

Le dispositif 105 comporte également un moyen de stockage 108 tel que par exemple un disque dur. Il comporte aussi un lecteur 109 de disque amovible 110. Ce disque 110 peut être une disquette, un CD-ROM ou un DVD-ROM, par exemple. Le disque 110 comme le disque 108 peuvent contenir des codes sources de programmes destinés à subir une partition selon l'invention ainsi que le ou les programmes mettant en œuvre l'invention qui, une fois lu par le dispositif 105, sera stocké dans le disque dur 108. Selon une variante, le programme permettant au dispositif de mettre en œuvre l'invention, pourra être stocké en mémoire morte 102 (appelée ROM sur le dessin). En seconde variante, le programme pourra être reçu pour être stocké de façon identique à celle décrite précédemment par l'intermédiaire du réseau de communication 113.

Ce même dispositif possède un écran 104 permettant de visualiser le code source du programme destiné à subir une partition ou pouvant servir d'interface avec l'utilisateur, qui peut ainsi paramétrer certains modes de traitement à l'aide du clavier 114 ou de tout autre moyen (souris par exemple).

L'unité centrale 100 (appelée CPU sur le dessin) exécute les instructions relatives à la mise en œuvre de l'invention, instructions stockées dans la mémoire morte 102 ou dans les autres éléments de stockage. Lors de la mise sous tension, les programmes de traitement stockés dans une mémoire non volatile, par exemple la ROM 102, sont transférés dans la mémoire vive RAM 103 qui contiendra alors le code exécutable de l'invention ainsi que des registres pour mémoriser les variables nécessaires à la mise en œuvre de l'invention.

De manière plus générale, un moyen de stockage d'information, lisible par un ordinateur ou par un microprocesseur, intégré ou non au dispositif, éventuellement amovible, mémorise un programme mettant en œuvre le procédé de partition de programme informatique.

Le bus de communication 101 permet la communication entre les différents éléments inclus dans le micro-ordinateur 105 ou reliés à lui. La représentation du bus 101 n'est pas limitative et notamment l'unité centrale 100 est susceptible de communiquer des instructions à tout élément du micro-

ordinateur 105 directement ou par l'intermédiaire d'un autre élément du micro-ordinateur 105.

En référence à la figure 2, un mode de réalisation de dispositif 200 de partition de programme informatique selon l'invention est destinée à réaliser une partition d'un programme informatique reçu sous forme de code source 210. Le dispositif 200 reçoit également en entrée une liste 220 des fonctions de bas niveau relatives à l'accès, à la lecture et à l'écriture d'informations. Ces informations sont par exemple organisées dans des fichiers informatiques ou dans des sources de données situées sur des sites de stockage. Ces fonctions de bas niveau sont par exemple les fonctions systèmes d'ouverture 221 et de fermeture de fichier, et les fonctions système d'écriture 222 et de lecture 223 d'informations dans un fichier.

Le dispositif selon l'invention comporte des moyens d'instrumentation 2010 d'un code source informatique.

Ces moyens sont par exemple constitués par un programme informatique de type profileur (profiler en anglais) tel que le programme « prof » du système d'exploitation UNIX. Quoi qu'il en soit, ces moyens d'instrumentation, permettent, à partie de règles prédéfinies, d'insérer ou de modifier des lignes de code dans le code source 210. Ces lignes de code seront décrites en référence aux figures 5a et 5b. Elles permettent en particulier, lors de leur exécution, d'effectuer des mesures qui permettront de calculer des données représentatives des transferts d'information. En outre, les appels aux fonctions de bas niveau du tableau 220 sont systématiquement remplacés par des appels à des fonctions spécifiques pour déterminer, en particulier, les sources et les cibles des transferts d'information. Ces fonctions spécifiques seront décrites ultérieurement en référence aux figures 6a, 6c et 6d.

Le dispositif 200 comporte aussi des moyens de compilation 2020 du programme instrumenté par les moyens d'instrumentation 2010. Ces moyens de compilation sont connus de l'homme du métier et ils ne seront pas décrits.

Le dispositif 200 comporte également des moyens d'exécution 2030 du programme instrumenté et compilé. Ces moyens comportent en particulier des moyens de commande et d'arrêt d'exécution d'un programme informatique, ainsi que des moyens de mesure de cette durée d'exécution. Ces moyens permettent aussi de cumuler les mesures réalisées par l'exécution du code instrumenté dans le tableau 700, qui sera décrit en référence aux figures 7a et 7b. De tels moyens peuvent en particulier être réalisés par des programmes informatiques similaires à ceux mis en œuvre dans les outils de déboguage et d'émulation de programmes informatiques.

Le dispositif 200 comporte en outre des moyens d'analyse 2040 des données représentatives cumulées dans le tableau 700 par les moyens d'exécution. Ces moyens d'analyse sont en particulier adaptés à construire un graphe d'appel et à déterminer les points de coupure de ce graphe permettant de minimiser le transfert d'informations échangées entre des sites distants. Ces moyens sont par exemple constitués par des circuits mettant en œuvre les procédés décrits ultérieurement en référence aux figures 10, 11 et 12.

Enfin, le dispositif 200 est adapté à fournir à l'utilisateur, par exemple sous forme d'un fichier, une liste 230 des points de coupures déterminés par les moyens d'analyse 2040. Ce fichier peut être enregistré par exemple sur le disque dur 108 de la figure 1.

La figure 3a représente un programme informatique 10 susceptible de subir une partition selon l'invention. Le programme 10 décrit ici est constitué de trois sous-programmes SP11, SP12 et SP13, tous situés sur un site de traitement 1. Les sous-programmes SP12 et SP13 accèdent directement à des informations situées sur des sites de stockage d'information 2 et 3, via un réseau de communication 4. Ces informations sont par exemple contenues dans des sources de données DB1 et DB2 situées physiquement dans chacun de ces sites. Le sous-programme SP11 faisant appel à ces sous-programmes SP12 et SP13, accède également à ces informations, mais de manière indirecte. Enfin, le sous-programme SP11 accède à des informations contenues dans une source de donnée DB3, située sur le site de traitement 1 du programme 10.

La figure 3b représente le programme 10 après qu'il a subi une partition conforme à l'invention. Par exemple, le sous-programme SP12 a été affecté au site 2, dans le but de réduire le temps d'exécution du programme 10. Ainsi les informations transférées entre le sous-programme SP12 et la source de données DB1, sont maintenant transférées localement au sein du site 2. En revanche, des informations transitent toujours sur le réseau 4, en particulier entre les sites 1 et 2 quand le sous-programme SP11 appelle le sous-programme SP12.

Concrètement, l'appel au sous-programme SP12 depuis le sous-programme SP11 a été transformé en appel distant. On dira dans ce cas qu'un point de coupure C1 a été inséré dans le sous-programme SP11 au niveau de l'appel au sous-programme SP12. De même un point de coupure C2 a été inséré dans le programme SP11 au niveau de l'appel au sous-programme SP13 qui a été affecté au site 3.

La recherche de ces points de coupures s'effectue en supposant que le programme doit toujours être lancé depuis le site d'exécution d'origine.

L'insertion manuelle de points de coupure est connue de l'homme du métier.

La figure 4 représente un mode de réalisation du procédé conforme à l'invention, comprenant des étapes E310 à E350.

Il comporte une première étape E310 d'instrumentation du code source 210 qui sera décrite en référence aux figures 5a et 5b.

La figure 5a est un exemple de code source 210, correspondant à des lignes L411 à L417 du code source du programme 10 de la figure 3a.

Dans l'exemple décrit ici, le sous-programme SP11 ne reçoit ni ne retourne aucun paramètre.

À la ligne L411, le sous-programme SP11 fait appel au sous-programme SP12 avec deux paramètres d'entrée : l'entier « 3 » et la chaîne de caractères « exemple ». La valeur retournée par le sous-programme SP12 est affectée à une variable de travail STR_TEMP de type chaîne de caractères qui est écrite dans la source de données DB3 à la ligne L413.

Dans l'exemple décrit ici, le sous-programme SP12 reçoit un premier paramètre d'entrée entier N et un deuxième paramètre d'entrée STR1 sous forme d'une chaîne de caractères. Dans l'exemple de l'appel par SP11 décrit précédemment, les paramètres N et STR1 prendront respectivement les valeurs 3 et « exemple ».

A la ligne L414, le sous-programme SP12 ouvre la source de données DB1. Cette opération s'effectue par l'exécution de la fonction de bas niveau OPEN 221 de la liste des fonctions de bas niveau 220 de la figure 2. Puis à la ligne L415, il réalise l'écriture de la chaîne de caractères STR1 correspondant à son deuxième paramètre d'entrée dans la source de données DB1. Cette opération se réalise par l'exécution de la fonction de bas niveau WRITE 222 de la liste des fonctions de bas niveau 220. Le résultat de cette opération sera dans cet exemple l'écriture de la chaîne de caractère « exemple » dans la source de données DB1.

De la même façon, à la ligne L416, le sous-programme SP12 lit une valeur dans la source de données DB1 par l'exécution de la fonction de bas niveau READ 223 de la liste des fonctions de bas niveau 220. Cette valeur lue, que nous supposerons être la chaîne de caractères « autre_exemple », est affectée à la variable STR2.

Cette valeur est ensuite retournée au sous-programme appelant, SP11 dans cet exemple, à la ligne L417.

Le sous-programme SP11 récupère ainsi la chaîne de caractères « autre_exemple » et l'affecte à la variable de travail STR_TEMP à la ligne L411.

A la ligne L412, le sous-programme SP11 ouvre la source de données DB3. Cette opération s'effectue par l'exécution de la fonction de bas niveau OPEN 221 de la liste des fonctions de bas niveau 220 de la figure 2. Puis à la ligne L413, il réalise l'écriture de la chaîne de caractères STR_TEMP dans la source de données DB3. Cette opération se réalise par l'exécution de la fonction de bas niveau WRITE 222 de la liste des fonctions de bas niveau 220. Le résultat de cette opération sera dans cet exemple l'écriture de la chaîne de caractère « autre_exemple » dans la source de données DB3.

La figure 5b représente le code source 210' correspondant à l'instrumentation du code source 210 de la figure 5a. Une fois compilée, l'exécution du programme instrumenté permettra d'obtenir les données représentatives de temps et de quantités de transfert d'information.

Tout d'abord une première ligne L421 est insérée au début de chacun des sous-programmes du programme 10. Cette première ligne affecte à une variable de travail SP_APPELANT la référence du programme appelant. Cette référence est par exemple obtenue à partir de la pile d'exécution du sous-programme en cours d'instrumentation. L'utilisation d'une telle pile d'exécution est connue de l'homme du métier, notamment pour le déboguage de programmes informatiques, et ne sera donc pas décrite.

Puis, une deuxième ligne L422 est insérée à la suite de la ligne L421. Cette ligne L422 appelle la fonction INC_APPELS avec deux paramètres d'entrée. Le premier paramètre d'entrée correspond à la variable SP_APPELANT affectée à la ligne L421. Le deuxième paramètre correspond à la référence du sous-programme en cours d'instrumentation, soit « SP11 » lors de l'instrumentation du sous-programme SP11 et « SP12 » lors de l'instrumentation du sous-programme SP12.

Pour chaque sous-programme ayant des paramètres d'entrée, l'instrumentation insère ensuite une ligne L425. Cette ligne L425 permet d'une part d'obtenir la taille QE des informations reçues en entrée par le sous-programme en cours d'instrumentation, et d'autre part d'effectuer, par l'appel d'une fonction CUMUL, le cumul de la taille de ces informations reçues en entrée tout au long de l'exécution du sous-programme instrumenté.

Comme illustré à la figure 5b, cette ligne n'est pas insérée pour le sous-programme SP11, qui n'a pas de paramètre d'entrée, et le cumul de la taille de ces informations est 0 pour ce sous-programme. Pour le sous-programme SP12 en revanche, la variable QE aura pour valeur la taille des paramètres d'entrée de SP12, c'est à dire la somme de la taille de l'entier N et de la taille de la chaîne de caractères STR1. La fonction CUMUL, quant à elle, reçoit quatre paramètres d'entrée, correspondants respectivement à la référence du sous-programme appelant, à la référence du sous-programme

appelé, à la variable de travail QE et à l'indice de la colonne du tableau 700 où le cumul doit être stocké. Dans le cas de paramètres d'entrée, comme à la ligne L425, ces données sont stockées dans la colonne ENTREE du tableau 700, comme décrit ultérieurement.

De même, pour chaque sous-programme ayant des paramètres de sortie, l'instrumentation insère, en fin du code source du sous-programme, une ligne L429. Cette ligne L429 permet d'une part d'obtenir la taille QS des informations renvoyées par le sous-programme en cours d'instrumentation et, d'autre part, via la fonction CUMUL, d'effectuer le cumul de la taille de ces informations tout au long de l'exécution du sous-programme instrumenté. Pour le cas des paramètres de sortie, le cumul s'effectue dans la colonne SORTIE du tableau 700.

Comme illustré à la figure 5b, cette ligne n'est pas insérée pour le sous-programme SP11 qui n'a pas de paramètre de sortie. Pour le sous-programme SP12 en revanche, la variable QS aura pour valeur la taille des paramètres de sortie de ce sous-programme, c'est à dire la taille de la chaîne de caractères STR2.

Enfin, l'instrumentation remplace chaque appel à une fonction de bas niveau de la liste 220 de la figure 2 par un appel à une fonction spécifique correspondante. Ces fonctions spécifiques, qui seront décrites en référence aux figures 6a, 6c et 6d, permettent d'une part de recenser quelles sont les informations transférées depuis ou vers un site de stockage distant, et d'autre part d'obtenir la quantité de ces informations. Ces fonctions spécifiques gardent les paramètres d'entrée et de sortie de la fonction de bas niveau d'origine.

Par exemple, chaque appel à la fonction de bas niveau OPEN 221 est remplacé par un appel à une fonction XOP 621.

De façon identique, une fonction de bas niveau CLOSE, non représentée ici, aurait été remplacée à l'étape d'instrumentation par une fonction XCL.

Ainsi les lignes L412, L413, L414, L415 et L416 de la figure 5a sont remplacées respectivement par les lignes L423, L424, L426, L427 et L428 de la figure 5b.

En référence à nouveau à la figure 4, l'étape d'instrumentation E310 du programme est suivie de l'étape E320. L'étape E320 est la compilation du code source instrumenté à l'étape E310. La compilation est classique et ne sera donc pas décrite ici. A l'issue de cette étape, le programme 10 instrumenté pourra être exécuté et les données représentatives du transfert d'information seront obtenues.

Le programme se branche ensuite à l'étape E322 de déclenchement d'un compteur TIMER. Ce compteur TIMER permet de mesurer une durée d'exécution du programme instrumenté, comme précisé ci-dessous.

Les étapes E324, E326 et E328 contrôlent l'exécution du programme 10 précédemment instrumenté et compilé. Cette exécution est démarrée à l'étape E324 et se poursuit tant que le résultat du test de l'étape E326, comparant la valeur de la variable TIMER à une constante DUREE est négatif. Lorsque la variable TIMER devient supérieure à cette constante, le résultat du test E326 devient positif et l'exécution du programme 10 instrumenté et compilé est stoppée à l'étape E328.

L'exécution du programme 10 instrumenté et compilé s'effectue à l'étape E327 tant que le résultat du test E326 est négatif. Cette étape E327 permet d'obtenir les données représentatives du transfert d'informations. Ces données sont regroupées dans le tableau 700 de la figure 7a, comme décrit ultérieurement.

Nous allons décrire ici comment ces données sont obtenues dans le cas de l'appel du sous-programme SP12 par le sous-programme SP11. La première ligne du sous-programme SP12 instrumenté exécutée est la ligne L421 (voir figure 5b). Lors de l'exécution de celle ligne, la référence du sous-programme appelant SP11 est affectée à la variable de travail SP_APPELANT.

Puis, le procédé exécute l'instruction de la ligne L422, qui fait appel à une fonction INC_APPELS. Cette fonction incrémente le nombre d'appels du sous-programme appelant (dans ce cas le sous-programme SP11) vers le sous-programme appelé (dans ce cas le sous-programme SP12), dans le tableau 700.

Le tableau 700 regroupe les données représentatives du transfert d'information lors de l'exécution du programme informatique 10 instrumenté à l'étape E310. Le tableau 700 comporte plusieurs lignes, chaque ligne regroupant les données représentatives du transfert d'information échangées entre un sous-programme appelant et un sous-programme appelé. Plus particulièrement, lors du premier appel d'un sous-programme appelant vers un sous-programme appelé, une ligne est créée dans le tableau 700. Le procédé affecte la première case de cette ligne avec la référence du sous-programme appelant et la deuxième case avec la référence du sous-programme appelé. La troisième case correspondant au nombre d'appels du sous-programme appelant vers le sous-programme appelé est initialisée à 1. Les cases des colonnes ENTREE et SORTIE destinées à contenir respectivement le cumul des quantités d'information échangées d'une part du sous-programme appelant vers le sous-programme appelé et d'autre part depuis le sous-programme appelé vers le sous programme appelant sont alors initialisées à 0. De même, la sixième et dernière colonne du tableau 700 contenant le cumul des temps de transfert des informations échangées entre le sous-programme appelant et le sous-programme appelé, est initialisée à 0. Bien entendu, les données cumulées dans le tableau 700, le sont pendant toute la durée d'exécution du programme instrumenté, c'est à dire tant que le résultat du test E326 est négatif.

Au cours de la première exécution de la ligne L422, une ligne 710 du tableau 700 a donc été créée, cette ligne contenant dans ses trois premières cases les valeurs SP11, SP12 et 1.

Puis, lors de l'exécution de la ligne L425, la variable de travail QE prend pour valeur la somme des tailles des informations reçues en entrée par le sous-programme SP12. Dans ce cas particulier, les paramètres d'entrée lors de l'appel par le sous-programme SP11 sont l'entier 3 et la chaîne de caractères « exemple ». En supposant qu'un entier est représenté par quatre octets et un caractère par un octet, la variable QE prend pour valeur (4 + (1*7)) soit 11.

Au cours de cette même étape, le sous-programme SP12 fait appel à une fonction CUMUL qui va maintenant être décrite brièvement. La fonction CUMUL reçoit en entrée la référence du sous-programme appelant, la référence du sous-programme appelé, une valeur à cumuler et l'indice de la colonne du tableau 700 dans laquelle la valeur doit être cumulée. La référence du sous-programme appelant, ici SP11, et du sous-programme appelé, ici SP12, permettent de déterminer la ligne du tableau 700 dans laquelle la valeur doit être cumulée : il s'agit ici de la ligne 710. La variable QE est donc cumulée avec la valeur contenue dans la case de la colonne ENTREE de la ligne 710.

Le sous-programme SP12 instrumenté exécute ensuite la ligne L426. L'exécution de cette ligne fait appel à la fonction XOP 621 qui va maintenant être décrite en référence à la figure 6a.

La première étape E622 de la fonction XOP 621 évalue si la source de données reçue en paramètre d'entrée est locale, c'est à dire située sur un site identique au site de traitement du sous-programme en cours d'exécution ou si, au contraire, elle est située sur un site distant.

Dans le cas de transfert d'information sur un réseau de type TCP/IP, cette évaluation est réalisée par exemple en comparant l'adresse électronique de ce site, obtenu par analyse de son adresse IP, avec l'adresse électronique du site de traitement.

L'étape E622 stocke le résultat de cette évaluation dans un tableau S à deux colonnes, représenté figure 6b. Pour chaque source de données accédée, une ligne est ajoutée au tableau S. La première case de cette ligne contient la référence de la source de données et la deuxième case la valeur « L » dans le cas où la source de données est située sur un site local et la valeur « D » dans le cas contraire.

Ainsi, l'exécution de la ligne L423 du sous-programme SP11 instrumenté remplit la ligne 901 du tableau S de la figure 6b. De même, la ligne 902 sera remplie par l'exécution de la ligne L426 du sous-programme SP12.

A l'issue de l'étape E622, la fonction XOP exécute un test E623 utilisant le tableau S de la figure 6b pour déterminer si la source de données passée en paramètre est locale ou distante. Dans le cas d'une source de

données locale, le résultat du test E623 est négatif et le programme se branche à l'étape E627, décrite ultérieurement.

Dans le cas où la source de données est distante, le résultat du test E623 est positif et le programme XOP exécute alors deux étapes E624 et E625 qui sont respectivement analogues aux instructions des lignes L421 et L422 de la figure 5b. Ces étapes ajoutent des données au tableau 700 de la figure 7a. Dans l'exemple décrit ici, la ligne 720 du tableau 700 est créée. La première case de cette ligne 720 reçoit alors la référence SP12 du sous-programme ayant appelé la fonction XOP, la deuxième case reçoit la référence DB1 de la source de données et la troisième case reçoit la valeur 1 pour le premier accès à la source de donnée DB1 depuis le sous-programme SP12.

La fonction XOP exécute alors une étape E626 analogue à l'étape de la ligne L425 de la figure 5b. Au cours de cette étape, la variable de travail QE prend pour valeur la somme des tailles des informations reçues en entrée par la fonction XOP. Dans ce cas particulier, les paramètres d'entrée se limitent à la référence « DB1 ». En supposant que la taille de la référence « DB1 » est 3 octets, la variable QE a pour valeur 3.

La valeur de la variable QE est ensuite cumulée à la valeur contenue dans la case de la colonne ENTREE de la ligne 720 du tableau 700. La valeur de cette case initialement nulle est maintenant 3.

La fonction XOP exécute alors à l'étape E627 l'instruction OPEN (DB1) correspondant à l'étape originale de la ligne L414 du programme SP12 de la figure 5a.

A l'issue de l'exécution de la ligne L426 le sous-programme SP12 instrumenté de la figure 5b exécute la ligne L427. Cette étape fait appel à la fonction XWRT 631 qui va maintenant être décrite en référence à la figure 6c.

La fonction XWRT a pour paramètres d'entrée d'une part la référence d'une source de données et d'autre part des informations à écrire dans cette source de données.

A l'étape E632, la fonction XWRT exécute un test au cours duquel est vérifié dans le tableau S de la figure 6b si la source de données est locale ou distante. Dans le cas d'une source de données locale, le résultat du test

E632 est négatif et le programme se branche à l'étape E636, où est appelée la fonction de bas niveau WRITE 222 du tableau 220, ce qui correspond à l'instruction originale de la ligne L413 de la figure 5a.

Dans le cas où la données est distante, le résultat du test E632 est positif et la fonction XWRT exécute alors deux étapes E633, et E634, respectivement analogues aux étapes E624 et E625 de la figure 6a. Ces étapes viennent compléter le tableau 700 de la figure 7a. Dans l'exemple décrit ici, les première et deuxième cases de la ligne 720 du tableau 700 étant déjà remplies avec les références SP12 et DB1, elles ne sont pas modifiées. En revanche, la valeur contenue dans la troisième case est incrémentée d'une unité et devient 2.

Le programme XWRT exécute alors une étape E635 analogue à l'étape de la ligne L425 de la figure 5b. Au cours de cette étape, la variable de travail QE prend pour la taille totale des paramètres d'entrée de la fonction XWRT. Dans ce cas particulier, les paramètres d'entrée de cette fonction lors de son appel par le sous-programme SP12 sont la chaîne de sept caractères « exemple » et la référence « DB1 ». En supposant que la taille de la référence « DB1 » est 3 octets, la variable QE prend pour valeur ((1*7)+3) soit 10.

La valeur de la variable QE est ensuite cumulée à la valeur contenue dans la case de la colonne ENTREE de la ligne 720 du tableau 700. Cette valeur devient donc 13.

A l'issue de l'exécution de la ligne L427, le sous-programme SP12 instrumenté exécute la ligne L428. Cette ligne contient un appel à la fonction XRD 641 représentée figure 6d et qui va maintenant être décrite.

La fonction XRD a pour paramètre d'entrée la référence d'une source de données dans laquelle des informations doivent être lues.

A l'étape E642, la fonction XRD exécute un test au cours duquel est vérifié dans le tableau S de la figure 6b si la source de données est locale ou distante. Dans le cas d'une source de données locale le résultat du test E642 est négatif et la fonction se branche à l'étape E649 où est appelée la fonction de bas niveau READ 223 du tableau 220, ce qui correspond à l'instruction originale de la ligne L416 de la figure 5a.

Dans le cas où la source de données est distante, le résultat du test E642 est positif et le programme XRD exécute alors deux étapes E643 et E644, respectivement analogues aux étapes E624 et E625 de la figure 6a. Ces étapes viennent compléter le tableau 700 de la figure 7a.

Dans l'exemple décrit ici, les première et deuxième colonnes de la ligne 720 du tableau 700 contiennent déjà les références SP12 et DB1 et ne sont pas modifiées. La valeur contenue dans la troisième colonne est incrémentée d'une unité et devient 3.

La fonction XRD exécute alors une étape E645 analogue à l'étape de la ligne L425 de la figure 5b. Au cours de cette étape, la variable de travail QE prend pour valeur la taille des informations reçues en entrée par XRD. Dans ce cas particulier, le paramètre d'entrée lors de l'appel par le sous-programme SP12 est la référence « DB1 », et la valeur 3 est cumulée à la valeur déjà contenue dans la colonne ENTREE de la ligne 720 du tableau 700. Cette dernière valeur devient donc 16.

Puis, au cours de l'étape E646, la source de données DB1 est lue par exécution de la fonction de bas niveau READ 223 du tableau 220 et la valeur retournée est affectée à une variable de travail STR_TEMP2. Nous supposerons que ces informations lues correspondent à la chaîne de caractère « autre_exemple », de taille 13 octets. Ces informations seront celles qui seront retournées par la fonction XRD au sous-programme appelant. Cette étape E646 correspond à l'instruction originale de la ligne L416 de la figure 5a.

Ensuite, au cours de l'étape E647, de façon similaire à l'étape E645 déjà décrite, la taille des informations devant être retournées par la fonction XRD, soit 13 octets, est cumulée à la valeur contenue dans la case de la colonne SORTIE de la ligne 720 du tableau 700. Cette dernière valeur était jusqu'alors nulle et devient 13.

Enfin, la chaîne de caractères « autre_exemple » lue à l'étape E646, est retournée à l'étape E648 et affectée à la variable de travail STR2 au cours de l'exécution de la ligne L428 du sous-programme SP12 de la figure 5b.

Le sous-programme SP12 instrumenté à l'étape E310 exécute alors la ligne L429 de la figure 5b. Au cours de cette exécution, la variable de travail

QS prend pour valeur la taille des informations retournées par le sous-programme SP12, c'est-à-dire la taille de la variable de travail STR2, soit 13 dans cet exemple. La valeur de la variable QS est ensuite cumulée à la valeur contenue dans case de la colonne SORTIE de la ligne 710 du tableau 700. Cette dernière valeur était jusqu'alors nulle et devient 13.

Enfin, le sous-programme SP12 retourne, au cours de l'exécution de la ligne L417 la chaîne de caractères « autre_exemple » au sous-programme appelant SP11.

Nous allons maintenant décrire la sixième colonne du tableau 700 de la figure 7a. Cette colonne, contient les données représentatives du temps de transfert des informations distantes échangées, sur le réseau 4, soit entre deux sous-programmes (ligne 710), soit lors de l'accès à une source de données distante (ligne 720).

Ces données sont calculées, pour chacune des lignes du tableau 700 au cours d'une étape E329, consécutive à l'étape E328 (figure 4). De manière préférée, ces données tiennent compte de caractéristiques du réseau 4 telles que la latence et le débit.

Par exemple, pour une ligne du tableau 700, la sixième case comporte la valeur de la variable TPS_CUM, calculée de la manière suivante:

TPS_CUM = (NB_APP * LATENCE) + (QTE / DEBIT), où la variable NB_APP correspond à la valeur de la troisième colonne du tableau 700, la variable QTE correspond à la somme des valeurs des cases des colonnes ENTREE et SORTIE et où LATENCE et DEBIT sont des constantes.

Par exemple, si LATENCE égale 0.05 s/appel et DEBIT égale 10000 octets par seconde, on trouve les valeurs pour les lignes 710 et 720 respectivement égales à 0.0524s et 0.1529s.

En variante, d'autres caractéristiques du canal de transmission telles que le taux d'erreur, la charge moyenne, ou au moins une valeur dépendante du protocole de communication peuvent être utilisées pour calculer les données représentatives du temps de transfert des informations.

Par exemple :

TPS_CUM = (NB_APP / (1-ERR)) *

(LATENCE + ((QTE/NB_APP) + ENTETE)/ (DEBIT*(1-CHARGE)),

où ERR est une valeur mesurant le taux d'erreur, ENTETE est un nombre d'octets transmis en sus des informations, ce nombre étant dépendant du protocole de communication, et CHARGE une valeur représentative du taux d'utilisation du réseau de communication.

Le tableau 700' de la figure 7b est un tableau construit de manière identique à celui de la figure 7a pour l'ensemble des sous-programmes d'un autre programme informatique instrumenté qui servira d'exemple pour la suite de la description.

Par exemple, la ligne 710' du tableau 700' indique que le sous-programme SP2 a été appelé 300 fois depuis le sous-programme SP1, que le volume cumulé des données transférées depuis le sous-programme SP1 vers le sous-programme SP2 est de 5000 octets, que le volume cumulé des données transférées depuis le sous-programme SP2 vers le sous-programme SP1 est de 45000 octets et que le temps de transfert total de ces données est de 20s.

L'étape E329 de calcul des données représentatives des temps de transfert, est suivie par une étape E330 de création d'un graphe d'appel qui va maintenant être décrite en référence à la figure 8.

Le graphe d'appel de la figure 8 est obtenu de façon récursive en parcourant le tableau 700' de la figure 7b, à partir des sources de données.

Au cours d'une première phase, le procédé repère les lignes du tableau 700' de la figure 7b correspondant au transfert d'informations entre un sous-programme et une source de données. Pour chacune de ces lignes, il construit une branche du graphe d'appel entre un nœud représentatif de ce sous-programme et une représentation de la source de données. Cette branche est pondérée par la donnée représentative du temps de transfert des informations entre le sous-programme et la source de données lue dans la sixième colonne du tableau 700' de la figure 7b. De même, la référence de la

source de données est inscrite dans le nœud représentatif de ce sous-programme.

En parcourant le tableau 700' de la figure 7b de haut en bas, la première ligne faisant référence à un appel d'un sous-programme vers une source de données est la ligne 731. Un nœud 801 représentatif du sous-programme SP6 est donc créé, ainsi qu'une représentation 802 de la source de données DB1. Puis une branche 803, pondérée par la valeur 20s lue dans la sixième colonne de la ligne 731, est représentée. Enfin la référence « DB1 » est ajoutée au nœud 801.

Les sous-programmes et les sources de données ne sont représentés qu'une seule fois. Lorsque le procédé trouve, par exemple, un sous-programme déjà représenté, comme c'est le cas à la ligne 732 pour le sous-programme SP9, une autre branche 805 est créée avec pour origine le nœud 804 représentatif de ce sous-programme SP9. La référence de la source de données, DB2 dans ce cas, est ensuite ajoutée au nœud 804.

Dans une seconde phase, pour chaque sous-programme SP_i déjà représenté par un nœud, le procédé recherche dans le tableau 700' de la figure 7b, les sous-programmes SP_j appelant ce sous-programme, et crée une nouvelle branche pour représenter cet appel. Cette nouvelle branche est pondérée par la donnée représentative du temps de transfert des informations échangées entre le sous-programme SP_j et le sous-programme SP_i, lue dans la sixième colonne du tableau 700' de la figure 7b. Les références des sources de données contenues dans le nœud représentant le sous-programme SP_i, sont ajoutées au nœud représentatif du sous-programme SP_j.

Ce procédé est répété pour tous les sous-programmes représentés par un nœud dans le graphe d'appel. Quand tous les nœuds ont été traités, la construction du graphe est terminée.

A l'issue de l'étape E330, le programme se branche à une étape E340 d'établissement des points de coupure du programme 10 instrumenté. Dans l'invention décrite ici, « faire une coupure », consiste, pour un appel de sous-programme, à rendre cet appel distant et, pour un accès à une source de données distante, à conserver cet accès distant. L'objet de l'invention est de

réaliser des coupures de façon à minimiser le temps global de transfert de données entre les différents sites en affectant certains sous-programmes à des sites de traitement contenant les données distantes.

Nous allons maintenant expliciter le mécanisme de détermination des points de coupure. Dans un premier temps, nous considérons deux graphes d'appels simples représentés figures 9a et 9b.

Le graphe d'appel de la figure 9a représente le graphe d'appel pour un programme composé de 4 sous-programmes SP21, SP22, SP23 et SP24. Le sous-programme SP21 fait appel au sous-programme SP22 qui fait appel au sous-programme SP23 qui lui-même fait appel au sous-programme SP24. Le sous-programme SP24, quant à lui, transfère des informations depuis ou vers une source de données DB4 distante.

Comme précédemment décrit, les données représentatives du temps de transfert des informations échangées sont représentées sur les branches reliant les sous-programmes. Par exemple, le temps de transfert cumulé des informations échangées entre le sous-programme SP21 et le sous-programme SP22 est de 10s. Dans la configuration initiale, c'est à dire avant la coupure, les transferts entre les différents sous-programmes sont des transferts locaux. Seul le transfert entre le sous-programme SP24 et la source de données DB4 est un transfert distant.

Dans le cas par exemple, où une coupure 910 serait réalisée au niveau de l'appel du sous-programme SP24 par le sous-programme SP23, ce qui correspondrait à transférer le sous-programme SP24 sur le site de la source de données DB4, alors le transfert d'informations entre le sous-programme SP24 et la source de données DB4 deviendrait local au site de la source de données DB4. En revanche, le transfert d'information entre les sous-programmes SP23 et SP24 deviendrait un transfert distant. Le temps de transfert correspondant aux données échangées sur le réseau passerait ainsi de 100s à 1s. Il apparaît clairement que ce découpage du programme minimise le temps de transfert des informations entre les sites.

Considérons maintenant le graphe de la figure 9b. Il s'agit d'un graphe d'appel pour un programme composé de 4 sous-programmes SP31,

SP32, SP33 et SP34. Le sous-programme SP31 fait appel au sous-programme SP32 qui fait lui-même appel aux deux sous-programmes SP33 et SP34. Les sous-programmes SP33 et SP34 transfèrent des informations depuis ou vers une source de données DB5.

En suivant un raisonnement qui consiste simplement à réaliser des coupures sur les branches de moindre poids, il pourrait paraître préférable de réaliser d'une part une première coupure 920 entre les sous-programmes SP32 et SP33 et d'autre part une deuxième coupure 930 entre les sous-programmes SP32 et SP34. Dans cette configuration, seuls les transferts d'information entre les sous-programmes SP32 et SP33 d'une part et ceux entre les sous-programmes SP32 et SP34 d'autre part sont distants, ce qui correspond à un temps de transfert d'information cumulé de 5s + 5s soit 10s.

En revanche, en réalisant une coupure 940 entre le sous-programme SP31 et le sous-programme SP32, le seul transfert distant, une fois les sous-programmes SP32, SP33 et SP34 transférés sur le site de la source de données DB5, devient le transfert d'information entre les sous-programmes SP31 et SP32, dont le temps total de transfert d'informations est de 7s. Ce temps de transfert est le minimum que l'on puisse obtenir.

Afin de généraliser le raisonnement fait en référence à la figure 9b, nous allons maintenant expliquer comment, pour un sous-programme donné, calculer un temps de transfert des informations échangées entre ce sous-programme et les sources de données auxquelles il accède directement ou non.

Le temps de transfert minimum pour un nœud ND représentant un sous-programme est donné par la formule suivante :

$$T\_MIN(ND) = \sum_{FILS} (\min(T\_XFERT(ND, FILS), T\_MIN(FILS))$$

Autrement dit, ce temps de transfert minimum est la somme pour tous les fils du nœud, de la plus petite des deux valeurs constituées d'une part par le temps de transfert des informations échangées entre ce nœud et le fils et d'autre part par le temps de transfert minimum pour ce même fils.

Cette formule permet d'obtenir les valeurs T_MIN(ND) pour tous les nœuds ND de la figure 8, ces valeurs étant représentées dans les nœuds correspondants dans le graphe de la figure 10. Par convention, la valeur T_MIN pour une source de donnée est infinie.

Par exemple, pour le nœud SP5, qui a deux fils SP7 et SP8, on obtient :

$$T\_MIN(SP5) = \min (T\_XFERT(SP7, SP5), T\_MIN(SP7)) + \min (T\_XFERT(SP8, SP5), T\_MIN(SP8))$$

Bien entendu, pour les nœuds ND accédant directement à une ou plusieurs sources de données, T_MIN(ND) correspond à la somme des temps de transfert des données échangées entre le sous-programme représenté par ce nœud et l'ensemble de ces sources de données.

Soit, en référence aux branches 806 et 807 de la figure 8,

$$T\_MIN(SP7) = 10 \text{ et } T\_MIN(SP8) = 50.$$

De même, les valeurs T_XFERT(SP7, SP5) et T_XFERT(SP8, SP5) sont lues directement sur les branches 808 et 810 de la figure 8.

Finalement, $T\_MIN(SP5) = \min (100, 10) + \min (5, 50) = 10 + 5 = 15$

En appliquant cette méthode à chacun des nœuds du graphe de la figure 8, on obtient l'ensemble des valeurs T_MIN de la figure 10.

Nous allons maintenant expliquer, toujours en référence à la figure 10, comment sont déterminés les points de coupure. Comme précisé auparavant, les points de coupure sélectionnés selon l'invention, sont tels qu'ils minimisent les temps de transfert des informations distantes. Or, le temps de transfert total minimum des informations transférées de façon directe et indirecte depuis un sous-programme vers des sources de données distantes est donné par la valeur T_MIN correspondant à ce sous-programme.

Si l'on s'intéresse par exemple au nœud 1011 correspondant au sous-programme SP4, on s'aperçoit que si le sous-programme SP4 reste sur le site d'exécution d'origine, il n'est pas possible d'avoir un temps total de transfert des informations accédées directement ou indirectement par SP4 inférieur à 10s. Or le temps de transfert entre SP4 et SP2 est de 1s. Il est donc judicieux

de déplacer SP4 et bien entendu le sous-programme SP6 appelé depuis SP4, c'est-à-dire de réaliser une coupure entre SP2 et SP4.

La généralisation de cet exemple va maintenant être donnée en référence à la figure 11. La figure 11 représente les étapes d'un algorithme RECH_COUP permettant de déterminer les coupures devant être réalisées à partir d'un nœud ND du graphe d'appel, ce nœud ND étant donné en paramètre d'entrée.

Afin de déterminer l'ensemble des coupures du graphe d'appel de la figure 10, l'algorithme RECH_COUP est tout d'abord appelé avec la racine 1020 de ce graphe en paramètre d'entrée.

Au cours d'une première étape E1100, la variable de travail FILS prend pour valeur le premier fils FILS de ND, soit, dans cet exemple, le nœud 1025 représentatif du sous-programme SP2.

Au cours de l'étape suivante E1120, les valeurs T_MIN(FILS) et T_XFERT(ND,FILS) sont comparées. Ces valeurs se trouvent respectivement dans la représentation du nœud FILS et sur la branche reliant les nœuds ND et FILS de la figure 10. Quand ND et FILS représentent les noeuds 1020 et 1025, ces valeurs sont respectivement égales à 16s et 20s.

Lorsque la valeur T_MIN(FILS) est strictement inférieure à la valeur T_XFERT(ND,FILS), le résultat du test E1120 est positif. Ceci signifie qu'il ne doit pas être réalisé de coupure entre le nœud FILS et le nœud ND. L'étape E1120 est alors suivie par une étape E1140, au cours de laquelle l'algorithme de recherche des points de coupure RECH_COUP est mis en œuvre avec le nœud FILS comme paramètre d'entrée.

Dans l'exemple décrit ici, la valeur T_MIN(SP2) (16s) étant inférieure à T_XFERT(SP1, SP2), (20s), le résultat du test E1120 est positif et l'algorithme de recherche des points de coupures est mis en œuvre avec le nœud 1025 comme paramètre d'entrée.

L'étape E1140 est suivie par un test E1150 pour tester si tous les fils du nœud ND ont été traités. Si tel n'est pas le cas, le résultat du test E1150 est négatif et, à l'étape suivante E1160, la variable FILS prend pour valeur le fils suivant du nœud ND, soit, dans cet exemple, le nœud 1030, représentatif du

sous-programme SP3. Cette étape E1160 est suivie par l'étape E1120 déjà décrite.

Dès lors que la valeur T_MIN(FILS) est supérieure ou égale à la valeur T_XFERT(ND,FILS), le résultat du test E1120 est négatif. L'étape E1120 est alors suivie par une étape E1180 au cours de laquelle une coupure entre le nœud ND et le nœud FILS est ajoutée à la liste 230 (figure 2) des points de coupure.

En référence à nouveau à la figure 4, l'étape E340 est suivie de l'étape E350 d'affectation des sous-programmes à des sites distants.

La liste des points de coupure permet de déterminer quels sous-programmes doivent être déplacés vers un autre site : chaque sous-programme représenté par un nœud se trouvant sous un point de coupure doit être déplacé. Dans le cas du programme dont le graphe d'appel est représenté figure 10, les sous-programmes SP4, SP6, SP8, SP9 et SP20 doivent être déplacés.

Rappelons à cet effet, que le choix de ce site est effectué de manière à minimiser les transferts d'informations entre ce sous-programme et des sites distants. Pour les sous-programmes à déplacer qui n'accèdent qu'à une seule source de données, soit pour les sous-programmes SP4, SP6, SP8 et SP20 de la figure 10, le site choisi sera évidemment le site contenant cette source de données. Dans le cas des sous-programmes accédant à plusieurs sources de données, soit SP9 dans cet exemple, le site choisi sera le site contenant les sources de données pour lesquelles le transfert d'information cumulé entre ce sous-programme et ces sources de données est maximum.

La figure 12 représente les étapes d'un algorithme IDENT_SITE de détermination du site sur lequel un sous-programme doit être déplacé. Cet algorithme IDENT_SITE est appelé avec un paramètre d'entrée ND, correspondant au nœud représentatif de ce sous-programme dans le graphe d'appel de la figure 10. Nous allons maintenant décrire la figure 12 en utilisant l'exemple du sous-programme SP9 de la figure 10.

Lors d'une première étape E1200, le procédé effectue un test au cours duquel il teste si la référence ND reçue en paramètre d'entrée représente

une source de données. Dans l'affirmative, cette étape est suivie par l'étape E1290 au cours de laquelle l'algorithme renvoie l'adresse électronique du site contenant cette source de données. Dans le cas d'un réseau de type TCP/IP, cette adresse sera une adresse Internet.

Dans le cas où la référence ND représente un sous-programme, le résultat du test E1200 est négatif et cette étape est suivie par une étape E1210.

Au cours de cette étape, l'algorithme teste si le sous-programme représenté par le nœud ND accède à plusieurs sources de données. Il faut noter à cet effet que la liste des sources de données accédées par un sous-programme se trouve dans le nœud représentatif de ce sous-programme (figure 8). Si le sous-programme n'accède qu'à une seule source de données, le résultat du test E1210 est négatif. Au cours de l'étape suivante E1290, l'adresse électronique du site contenant cette source de données est renvoyée.

Dans l'exemple décrit ici, le sous-programme SP9 accède à deux sources de données DB1 et DB2 et le résultat du test E1210 est positif. Le test E1210 est alors suivi par une étape E1220 de création d'un tableau TPS_XFERT_SITE. Ce tableau est constitué d'autant de lignes que de sites contenant des sources de données accédées par le sous-programme et de deux colonnes. Dans la première colonne de ce tableau, on trouvera l'adresse électronique d'un site contenant une source de données accédée par ce sous-programme, et dans la deuxième colonne, initialisée à zéro, le cumul des temps de transfert des informations échangées entre ce sous-programme et les différentes sources de données de ce site.

L'étape E1220 est alors suivie par une étape E1230 au cours de laquelle la variable de travail FILS prend la valeur du premier fils FILS de ND, soit DB1 dans cet exemple.

Au cours de l'étape suivante E1240, l'algorithme de détermination d'un site IDENT_SITE est appelé avec la variable FILS comme paramètre d'entrée, soit DB1 dans cet exemple.

Dans ce cas, le résultat du test E1200 sera positif, puisque le paramètre d'entrée de cet algorithme est la référence d'une source de

données, et l'adresse électronique du site contenant la base de donnée DB1, par exemple « SITE_DB1 », sera renvoyée à l'étape E1290.

Cette adresse électronique est affectée à la variable de travail SITE_FILS au cours de l'étape E1240. Cette étape est suivie par l'étape E1250 au cours de laquelle le tableau TPS_XFERT_SITE créé à l'étape E1220 est complété. De façon plus précise, une ligne de ce tableau est utilisée pour cumuler les temps de transfert des informations échangées entre le sous-programme SP9 et le site SITE_DB1. La première fois que la variable SITE_FILS prend la valeur SITE_DB1, la première colonne d'une ligne de ce tableau reçoit la valeur SITE_DB1. La valeur du temps de transfert des informations échangées entre SP9 et DB1, T_XFERT(SP9,DB1) est ajoutée à la valeur contenue dans la case de la deuxième colonne de cette ligne. La valeur de cette case initialement nulle prend donc la valeur 5.

L'étape E1250 est suivie par un test E1260 pour tester si tous les fils du nœud ND ont été traités. Si tel n'est pas le cas, le résultat du test E1260 est négatif, et à l'étape suivante E1270, la variable FILS est affectée par le fils suivant du nœud ND, soit DB2 dans ce cas.

L'étape E1270 est alors suivie par l'étape E1240 déjà décrite où l'algorithme de détermination d'un site IDENT_SITE est appelé avec la valeur DB2 comme paramètre d'entrée. De façon identique, à l'étape E1250, une ligne est créée dans le tableau TPS_XFERT_SITE avec en première colonne l'adresse électronique du site contenant la source de données DB2, soit par exemple « SITE_DB2 », et en deuxième colonne la valeur T_XFERT(SP9,DB2), soit 20s.

Le résultat du test E1260 est cette fois positif, et cette étape est suivie par l'étape E1280. Cette étape retourne l'adresse électronique du site pour lequel la valeur du temps de transfert cumulé dans la deuxième colonne du tableau TPS_XFERT_SITE est maximale. Dans le cas du sous-programme SP9, l'étape E1280 renvoie l'adresse électronique « SITE_DB2 ».

A l'issue de l'étape E350 de la figure 4, l'utilisateur possède donc les adresses électroniques des sites sur lesquels certains sous-programmes

doivent être déplacés pour minimiser le temps de transfert global d'informations.

Bien entendu, la présente invention n'est nullement limitée aux modes de réalisation décrits et représentés, mais englobe, bien au contraire, toute variante à la portée de l'homme du métier.

REVENDICATIONS

1. Procédé de partition d'un programme informatique (10) situé sur un premier site de traitement (1), le programme comportant des sous-programmes (SP11, SP12, SP13) susceptibles de transférer des informations, caractérisé en ce qu'il comporte:

- une étape de détermination automatique (E310, E320, E327), pour au moins l'un desdits sous-programmes (SP11, SP12, SP13) , de données (QE, QS, TPS_CUM) représentatives du transfert d'au moins une partie des informations traitées par ledit sous-programme; et

- une étape d'affectation (E350) dudit sous-programme à un deuxième site de traitement (2, 3) en fonction desdites données (QE, QS, TPS_CUM).

2. Procédé de partition selon la revendication 1, caractérisé en ce que l'étape de détermination automatique des dites données représentatives comporte les sous-étapes suivantes :

- modification (E310) du code source (210) dudit programme informatique (10) ;

- compilation (E320) dudit code modifié (210') et génération d'un programme modifié ; et

- obtention desdites données représentatives (QE, QS, TPS_CUM) par au moins une exécution (E327) dudit programme modifié.

3. Procédé de partition selon la revendication 2, caractérisé en ce que ladite sous-étape de modification (E310) du code source (10) insère dans le code source (210) d'au moins un sous-programme (SP11, SP12, SP13) dudit programme informatique (10) :

- des premières lignes d'instructions (L421) permettant, lors de leur exécution (E327) d'obtenir et de mémoriser une référence (SP_APPELANT) d'un sous-programme appelant ledit sous-programme ; et

- des deuxièmes lignes d'instructions (L425, L429) permettant, lors de leur exécution (E327) d'obtenir et de mémoriser le cumul des données représentatives (QE, QS, TPS_CUM) des informations reçues ou transférées par ledit sous-programme.

4. Procédé de partition selon la revendication 2 ou 3, caractérisé en ce que ladite sous-étape de modification (E310) du code source (210) remplace, dans le code source d'au moins un sous-programme (SP11, SP12, SP13) dudit programme informatique (10), les appels à des fonctions de bas niveau (221, 222, 223) par des lignes d'instructions (L423, L424, L425) permettant, lors de leur exécution, d'obtenir et de mémoriser si les données transférées par ledit sous-programme sont situées sur un site de stockage distant (2, 3).

5. Procédé de partition selon l'une des revendications 2 à 4, caractérisé en ce que les dites données représentatives (QE, QS, TPS_CUM) sont obtenues de façon statistique, après au moins deux exécutions dudit programme informatique modifié.

6. Procédé de partition selon l'une quelconque des revendications 1 à 5, caractérisé en ce que lesdites données sont représentatives de la quantité (QE, QS) de ladite partie des informations traitées par ledit sous-programme.

7. Procédé de partition selon l'une quelconque des revendications 1 à 6, caractérisé en ce que lesdites données sont représentatives du temps de transfert (TPS_CUM) de ladite partie des informations traitées par ledit sous-programme.

8. Procédé de partition selon l'une quelconque des revendications 1 à 7, caractérisé en ce que les dites données (QE, QS, TPS_CUM) sont représentatives de caractéristiques d'un canal de transmission entre ledit premier site de traitement (1) et ledit deuxième site de traitement (2,3).

9. Procédé de partition selon la revendication 8, caractérisé en ce que lesdites données représentatives du canal de transmission sont choisies parmi la latence, la bande passante, le taux d'erreur, la charge moyenne du canal de transmission et au moins une valeur dépendante d'un protocole de communication de ladite partie des informations entre ledit premier site de traitement et ledit deuxième site de traitement (2, 3).

10. Procédé de partition selon l'une quelconque des revendications 1 à 9, caractérisé en ce que ladite étape d'affectation (E350) est réalisée de manière à minimiser, par analyse desdites données représentatives (QE, QS, TPS_CUM), le transfert d'informations entre le premier site de traitement (1) et ledit deuxième site de traitement (2, 3).

11. Dispositif de partition d'un programme informatique (10) situé sur un premier site de traitement (1), le programme comportant des sous-programmes (SP11, SP12, SP13) susceptibles de transférer des informations, caractérisé en ce qu'il comporte:

- des moyens de détermination automatique, pour au moins l'un desdits sous-programmes (SP11, SP12, SP13), de données (QE, QS, TPS_CUM) représentatives du transfert d'au moins une partie des informations traitées par ledit sous-programme; et

- des moyens d'affectation dudit sous-programme à un deuxième site de traitement (2, 3) en fonction desdites données (QE, QS, TPS_CUM).

12. Dispositif de partition selon la revendication 11, caractérisé en ce que les moyens de détermination automatique des dites données représentatives comporte :

- des moyens (2010) de modification du code source (210) dudit programme informatique (10) ;

- des moyens (2020) de compilation dudit code modifié (210') et de génération d'un programme modifié ; et

- des moyens (2030) adaptés à obtenir lesdites données représentatives (QE, QS, TPS_CUM) par au moins une exécution (E327) dudit programme modifié.

13. Dispositif de partition selon la revendication 12, caractérisé en ce que lesdits moyens (2010) de modification du code source (10) insèrent dans le code source (210) d'au moins un sous-programme (SP11, SP12, SP13) dudit programme informatique (10) :

- des premières lignes d'instructions (L421) permettant, lors de leur exécution (E327) d'obtenir et de mémoriser une référence (SP_APPELANT) d'un sous-programme appelant ledit sous-programme ; et

- des deuxièmes lignes d'instructions (L425, L429) permettant, lors de leur exécution (E327) d'obtenir et de mémoriser le cumul des données représentatives (QE, QS, TPS_CUM) des informations reçues ou transférées par ledit sous-programme.

14. Dispositif de partition selon la revendication 12 ou 13, caractérisé en ce que lesdits moyens de modification (2010) du code source (210) remplacent, dans le code source d'au moins un sous-programme (SP11, SP12, SP13) dudit programme informatique (10), les appels à des fonctions de bas niveau (221, 222, 223) par des lignes d'instructions (L423, L424, L425) permettant, lors de leur exécution, d'obtenir et de mémoriser si les données transférées par ledit sous-programme sont stockées sur un site de stockage distant (2, 3).

15. Dispositif de partition selon l'une des revendications 12 à 14, caractérisé en ce que lesdits moyens adaptés à obtenir lesdites données représentatives (QE, QS, TPS_CUM) opèrent de façon statistique, après au moins deux exécutions dudit programme informatique modifié.

16. Dispositif de partition selon l'une quelconque des revendications 11 à 15, caractérisé en ce qu'il est adapté à considérer des données représentatives de la quantité (QE, QS) de ladite partie des informations traitées par ledit sous-programme.

17. Dispositif de partition selon l'une quelconque des revendications 11 à 16, caractérisé en ce qu'il est adapté à considérer des données représentatives du temps de transfert (TPS_CUM) de ladite partie des informations traitées par ledit sous-programme.

18. Dispositif de partition selon l'une quelconque des revendications 11 à 17, caractérisé en ce qu'il est adapté à considérer des données (QE, QS, TPS_CUM) représentatives de caractéristiques d'un canal de transmission entre ledit premier site de traitement et ledit deuxième site de traitement (2, 3).

19. Dispositif de partition selon la revendication 18, caractérisé en ce qu'il est adapté à considérer des données représentatives du canal de transmission choisies parmi la latence, la bande passante, le taux d'erreur, la charge moyenne du canal de transmission et au moins une valeur dépendante d'un protocole de communication de ladite partie des informations entre ledit premier site de traitement et ledit deuxième site de traitement (2, 3).

20. Dispositif de partition selon l'une quelconque des revendications 11 à 19, caractérisé en ce que lesdits moyens d'affectation sont adaptés à minimiser, par analyse desdites données représentatives (QE, QS, TPS_CUM), le transfert d'informations entre le premier site de traitement (1) et le deuxième site de traitement (2, 3).

21. Dispositif de partition selon l'une quelconque des revendications 11 à 20, caractérisé en ce que les moyens de détermination automatique et d'affectation sont incorporés dans :

- une unité centrale (100),
- une mémoire morte (102) comportant les instructions relatives à la mise en œuvre de la partition; et
- une mémoire vive (103) comportant des registres adaptés à enregistrer des variables modifiées au cours de l'exécution desdites instructions.

22. Ordinateur caractérisé en ce qu'il comporte des moyens adaptés à mettre en œuvre le procédé selon l'une quelconque des revendications 1 à 10.

23. Ordinateur, caractérisé en ce qu'il comporte le dispositif selon l'une quelconque des revendications 11 à 21.

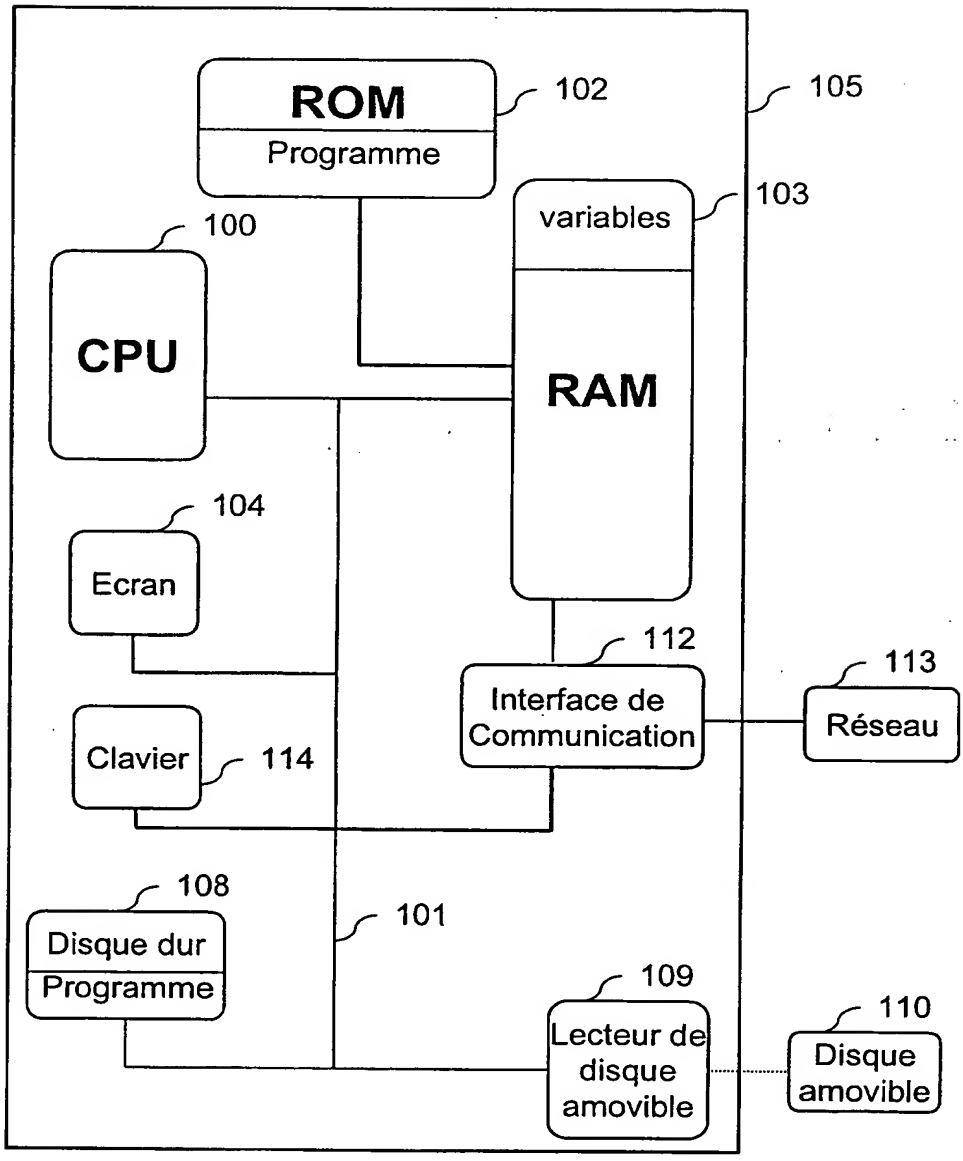

ROM
Programme
102
105
variables
103
100
CPU
RAM
104
Ecran
112
113
Interface de Communication
Réseau
Clavier
114
108
101
Disque dur
Programme
109
110
Lecteur de disque amovible
Disque amovible

FIG. 1

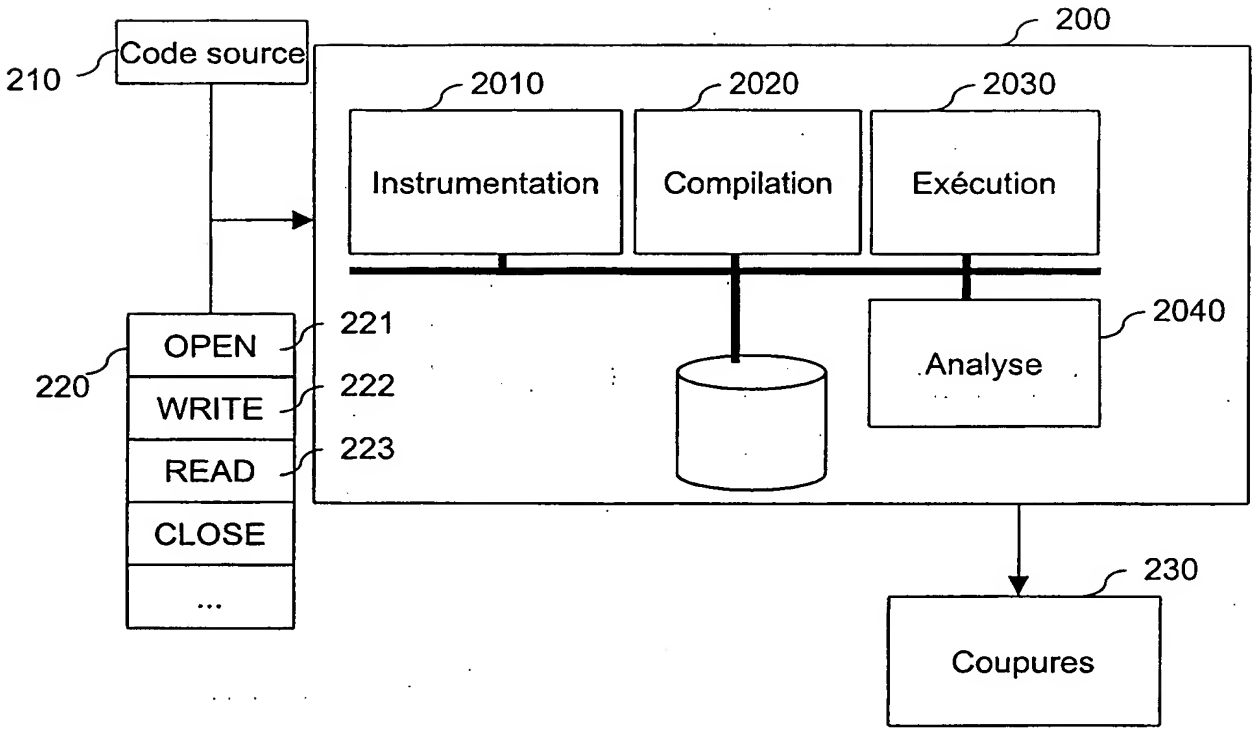

200
210
Code source
2010
2020
2030
Instrumentation
Compilation
Exécution
2040
Analyse
220
OPEN
221
WRITE
222
READ
223
CLOSE
...
230
Coupures

FIG. 2

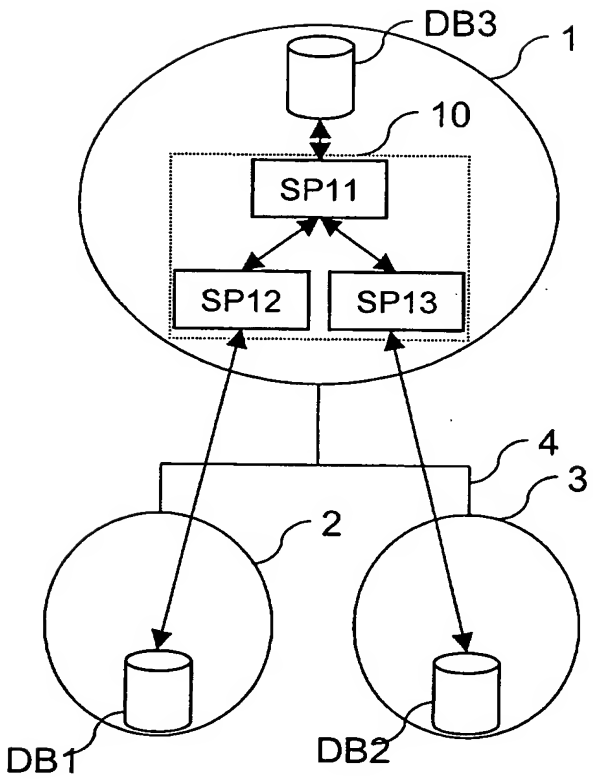

DB3
1
10
SP11
SP12
SP13
4
3
2
DB1
DB2

FIG. 3a

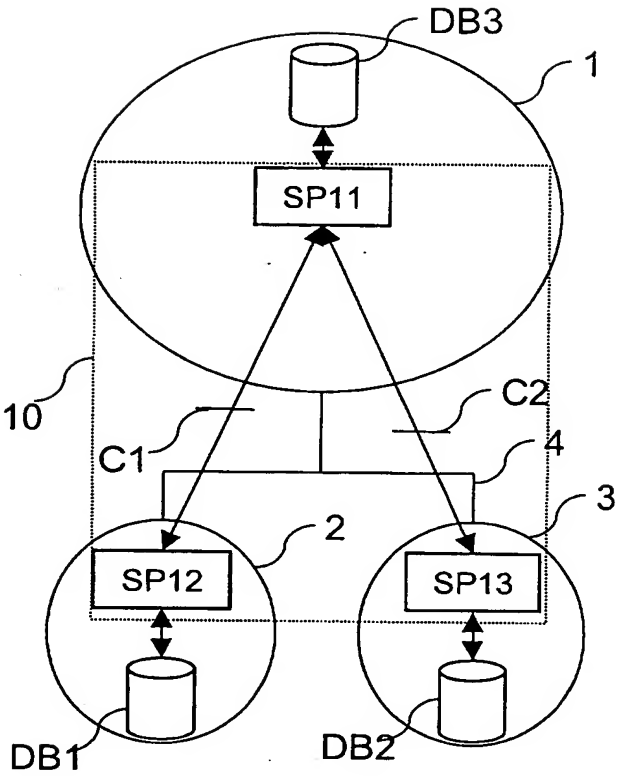

DB3
1
SP11
10
C2
C1
4
3
2
SP12
SP13
DB1
DB2

FIG. 3b

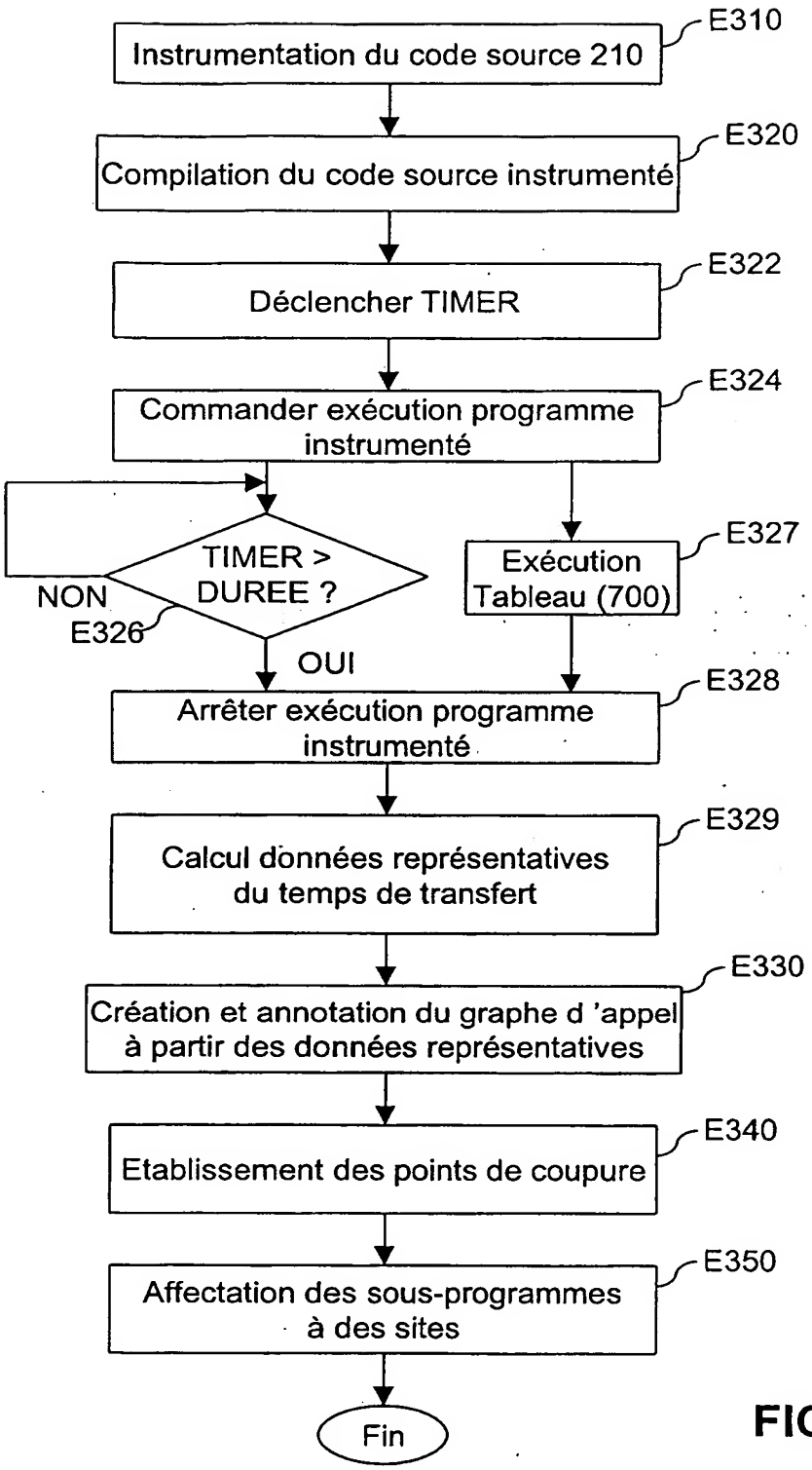

Instrumentation du code source 210
E310
Compilation du code source instrumenté
E320
Déclencher TIMER
E322
Commander exécution programme instrumenté
E324
TIMER > DUREE ?
NON
E326
OUI
Exécution Tableau (700)
E327
Arrêter exécution programme instrumenté
E328
Calcul données représentatives du temps de transfert
E329
Création et annotation du graphe d 'appel à partir des données représentatives
E330
Etablissement des points de coupure
E340
Affectation des sous-programmes à des sites
E350
Fin

FIG. 4

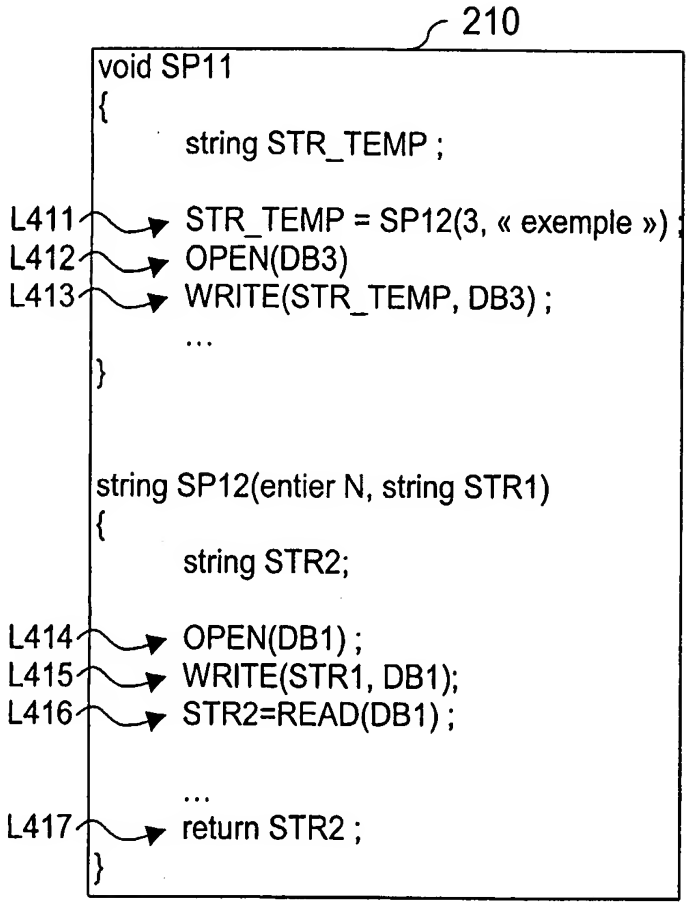


FIG. 5a

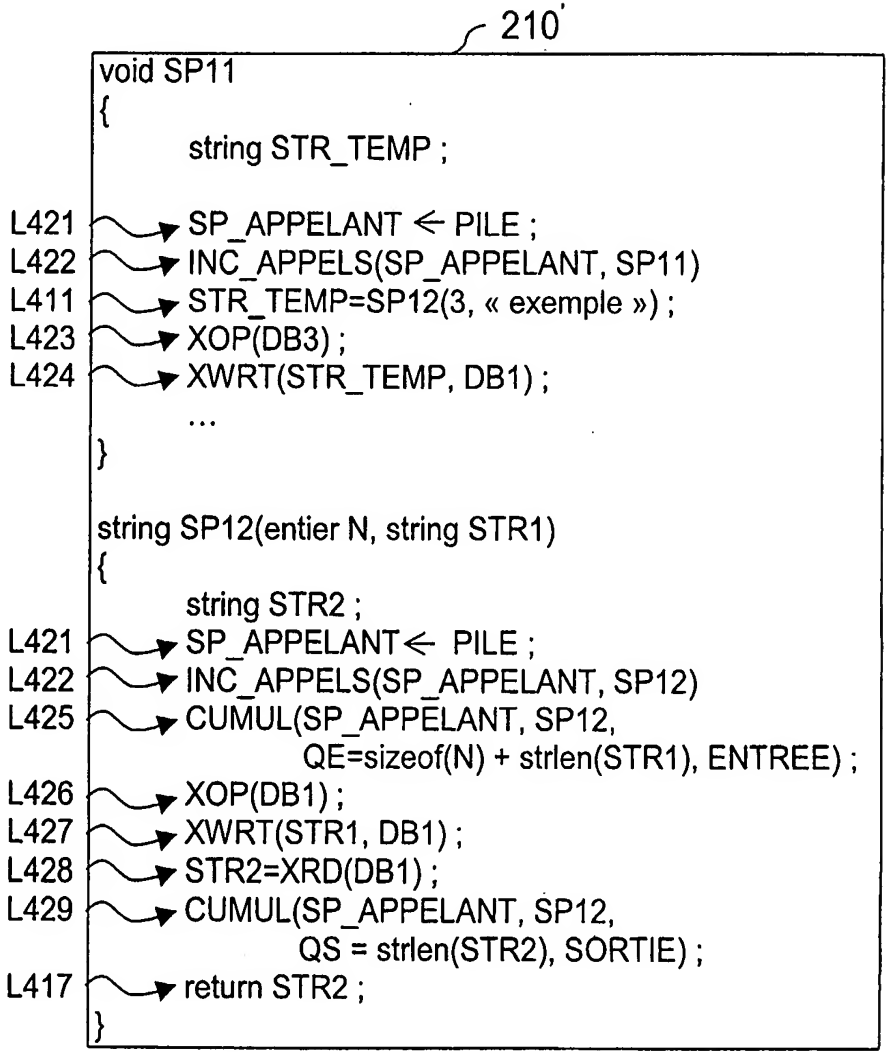


FIG. 5b

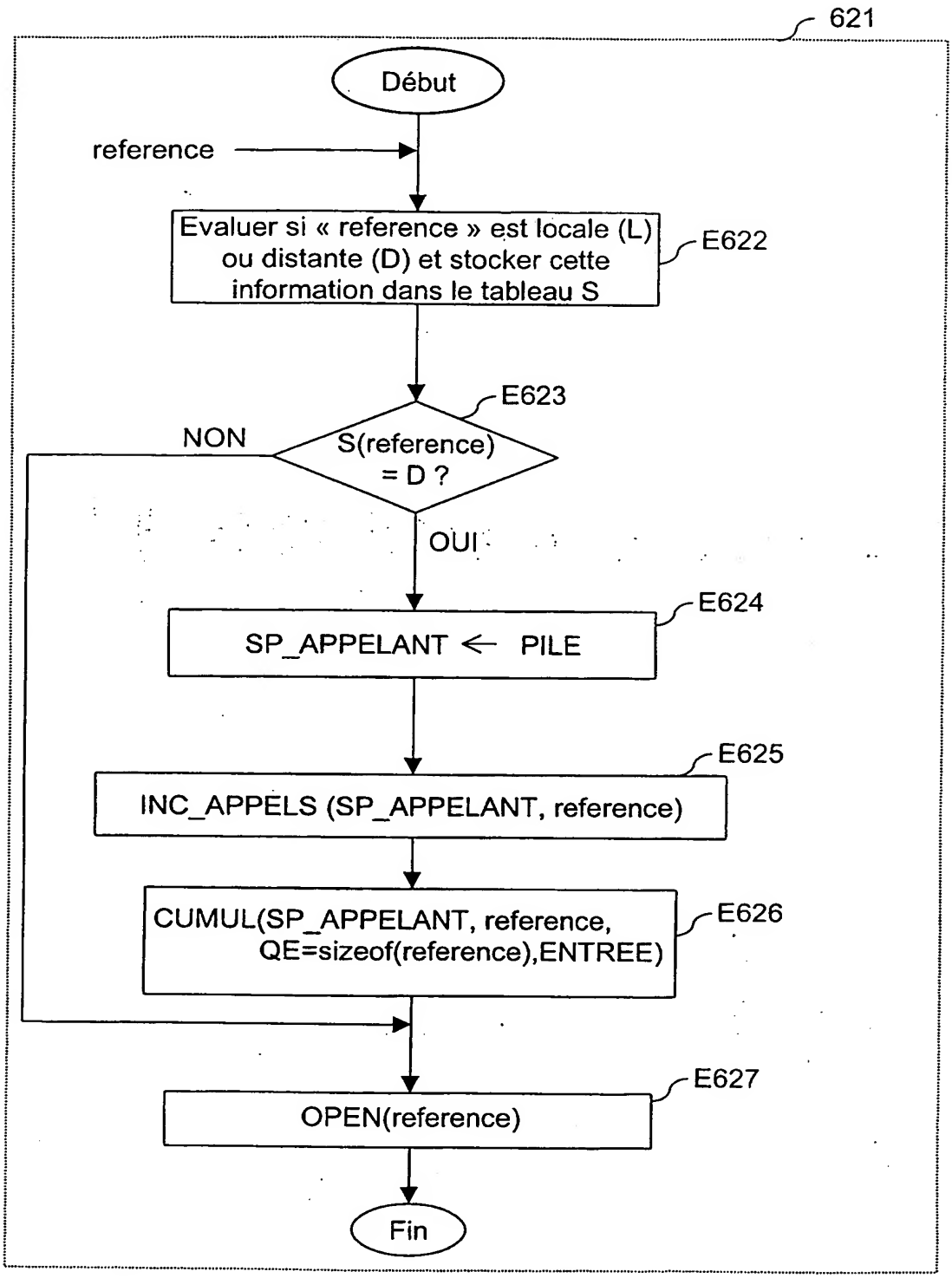

621
Début
reference
Evaluer si « reference » est locale (L) ou distante (D) et stocker cette information dans le tableau S
E622
E623
NON
S(reference) = D ?
OUI
E624
SP_APPELANT ← PILE
E625
INC_APPELS (SP_APPELANT, reference)
CUMUL(SP_APPELANT, reference, QE=sizeof(reference),ENTREE)
E626
E627
OPEN(reference)
Fin

FIG. 6a

S

Base de données	Local/Distant
DB3	L
DB1	D

901

902

FIG. 6b

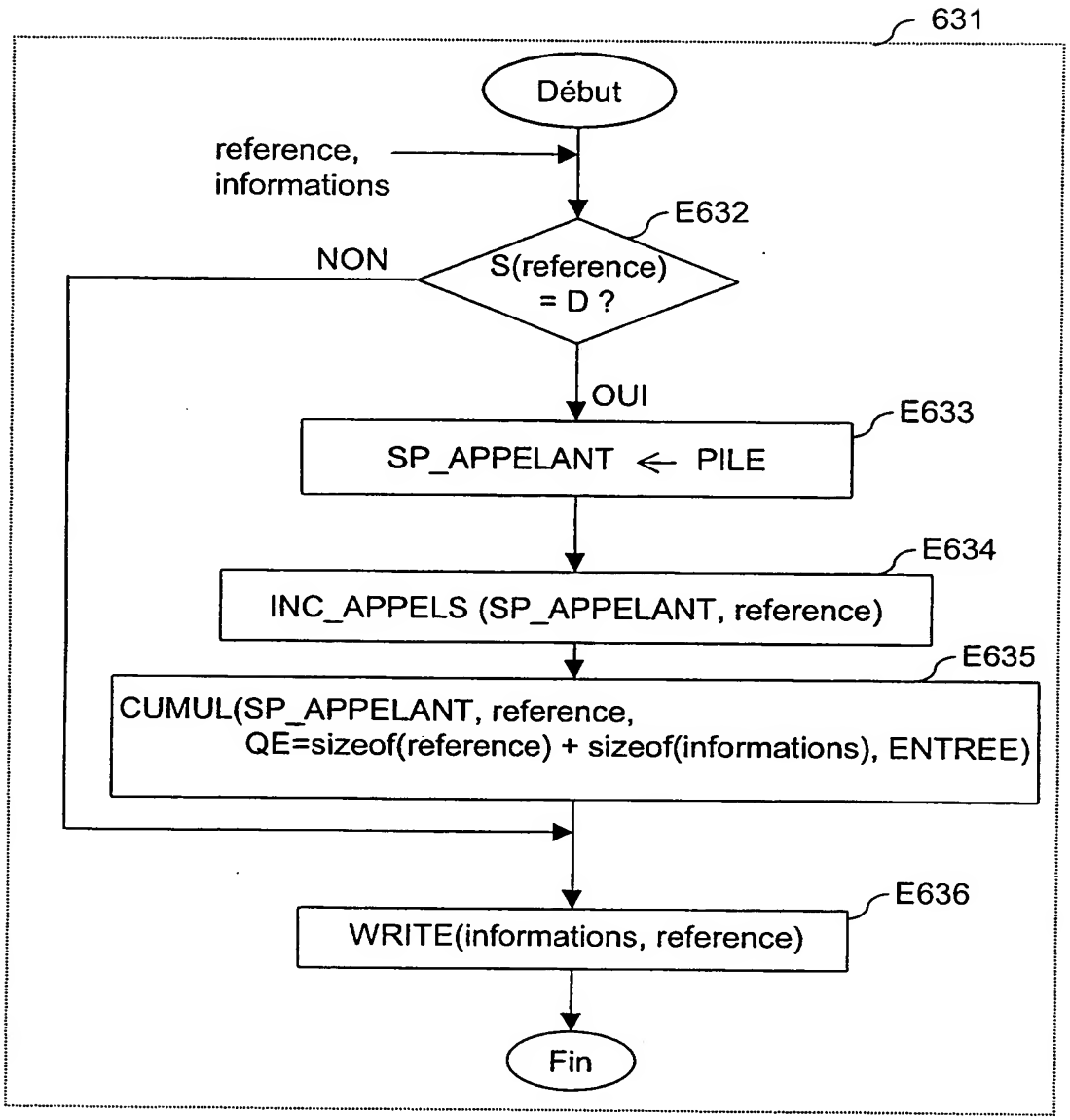


FIG. 6c

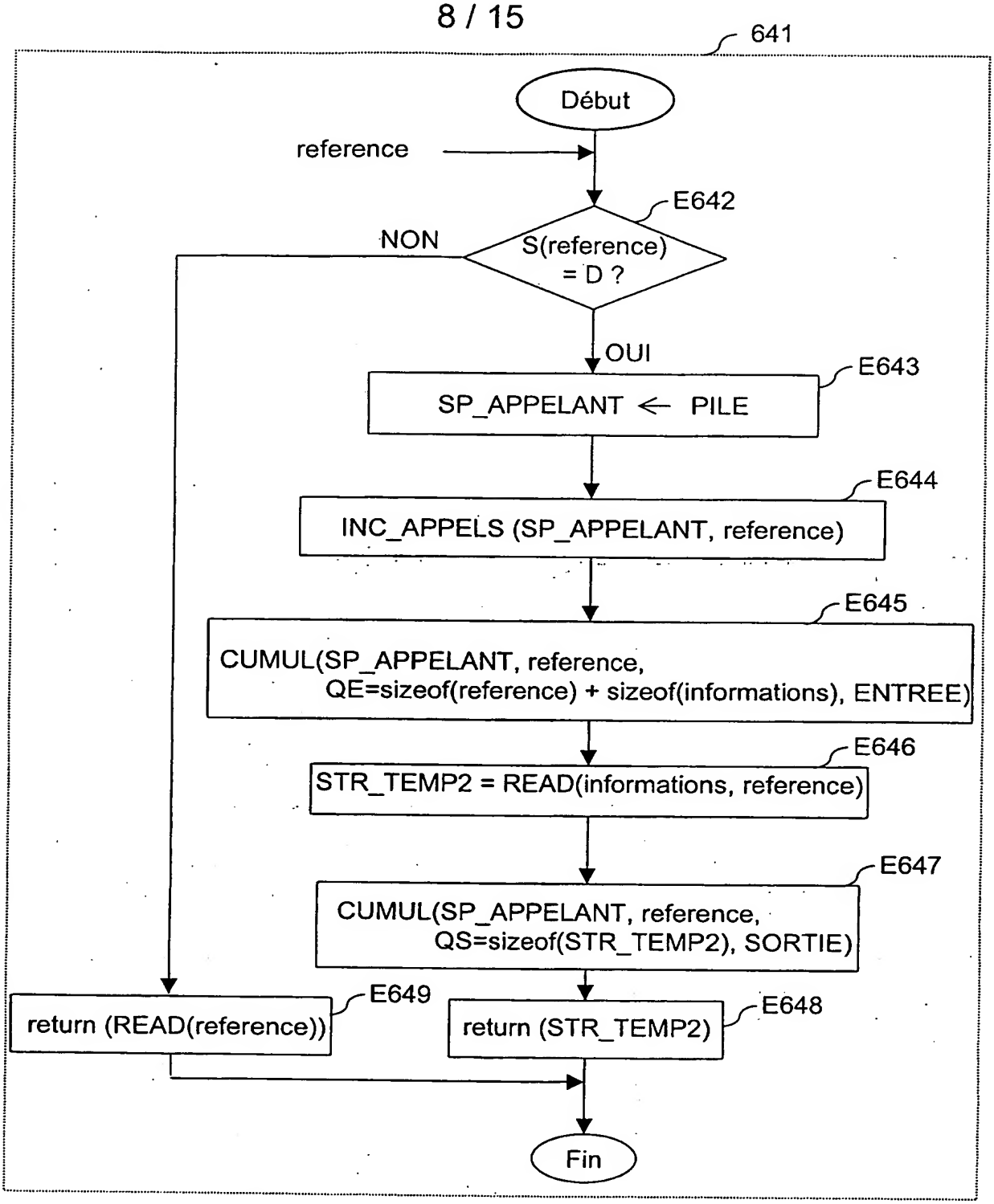

641
Début
reference
E642
S(reference) = D ?
NON
OUI
E643
SP_APPELANT ← PILE
E644
INC_APPELS (SP_APPELANT, reference)
E645
CUMUL(SP_APPELANT, reference, QE=sizeof(reference) + sizeof(informations), ENTREE)
E646
STR_TEMP2 = READ(informations, reference)
E647
CUMUL(SP_APPELANT, reference, QS=sizeof(STR_TEMP2), SORTIE)
E649
return (READ(reference))
E648
return (STR_TEMP2)
Fin

FIG. 6d

700

Sous-programme Appelant	Sous-programme Appelé	Nombre D'appels	Quantité d'information transférée de l'appelant vers l'appelé (octets)	Quantité d'information transférée de l'appelé vers l'appelant (octets)	Somme des temps de transfert (s)
SP11	SP12	1	11	13	0,0524
SP12	DB1	3	16	13	0,1587

710

720

ENTREE

SORTIE

FIG. 7a

700'

Sous-programme Appelant	Sous-programme Appelé	Nombre D'appels	Quantité d'information transférée de l'appelant vers l'appelé (octets)	Quantité d'information transférée de l'appelé vers l'appelant (octets)	Somme des temps de transfert (s)
SP1	SP2	300	5000	45000	20
SP1	SP3	550	5000	20000	30
SP2	SP4	18	800	200	1
SP2	SP5	1600	5000	150000	100
SP3	SP9	150	8000	17000	10
SP3	SP20	95	500	2000	5
SP4	SP6	190	2000	3000	10
SP5	SP7	1500	1000	249000	100
SP5	SP8	90	1000	4000	5
SP6	DB1	350	16000	9000	20
SP7	DB1	150	5000	20000	10
SP8	DB1	900	15000	35000	50
SP9	DB1	50	500	24500	5
SP9	DB2	350	5000	20000	20
SP20	DB2	1500	5000	245000	100
			ENTREE	SORTIE	

731

732

FIG. 7b

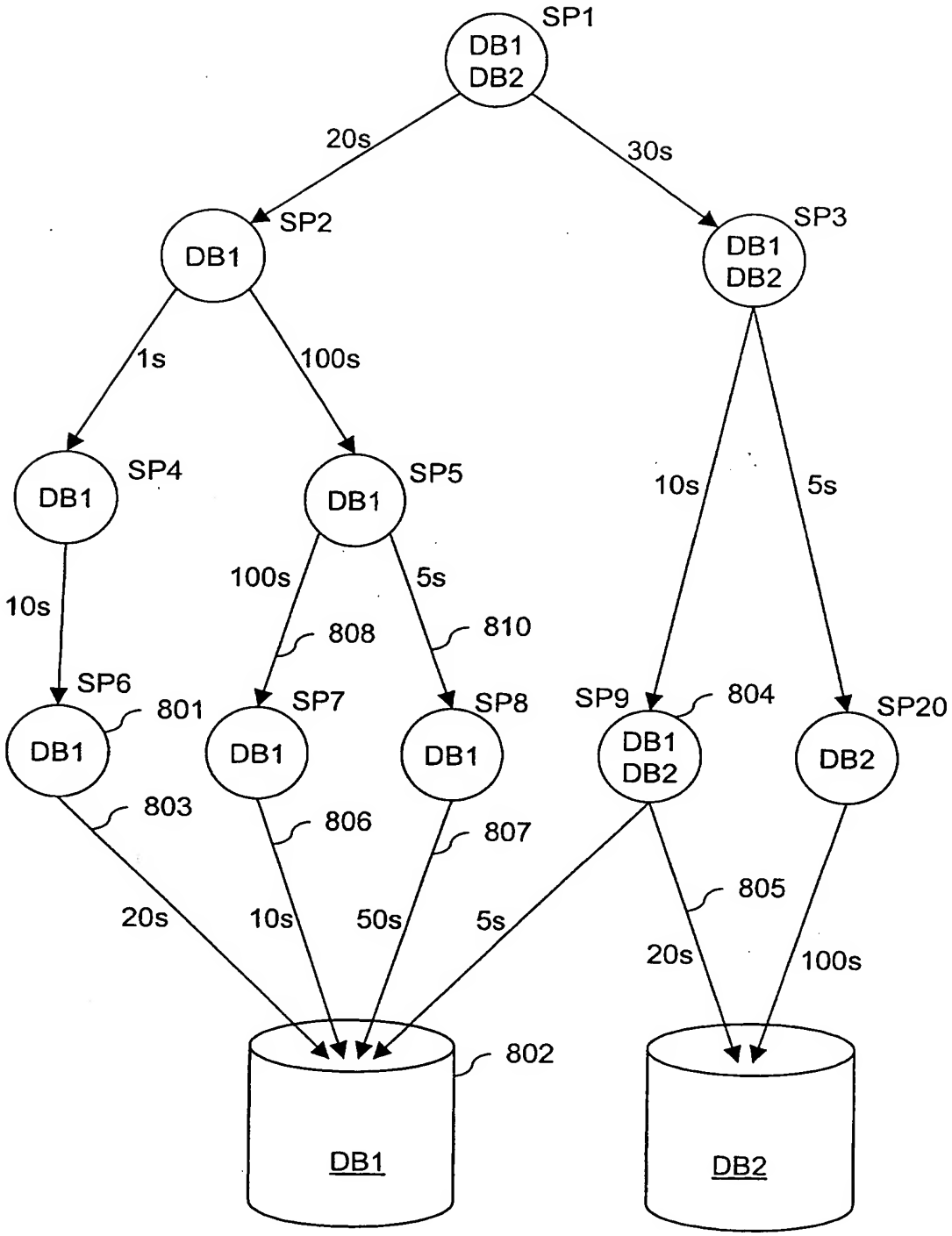

SP1
DB1
DB2
20s
30s
SP2
DB1
SP3
DB1
DB2
1s
100s
SP4
DB1
SP5
DB1
10s
5s
100s
5s
10s
808
810
SP6
801
SP7
SP8
SP9
804
SP20
DB1
DB1
DB1
DB1
DB2
DB2
803
806
807
805
20s
10s
50s
5s
20s
100s
802
DB1
DB2

FIG. 8

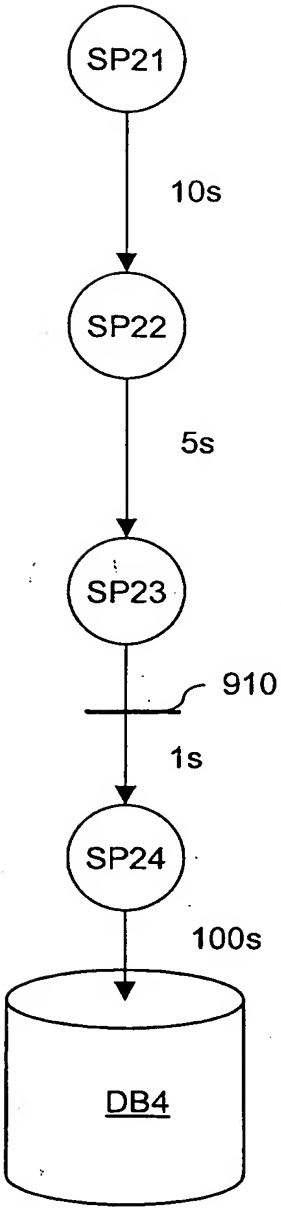

SP21
10s
SP22
5s
SP23
910
1s
SP24
100s
DB4

FIG. 9a

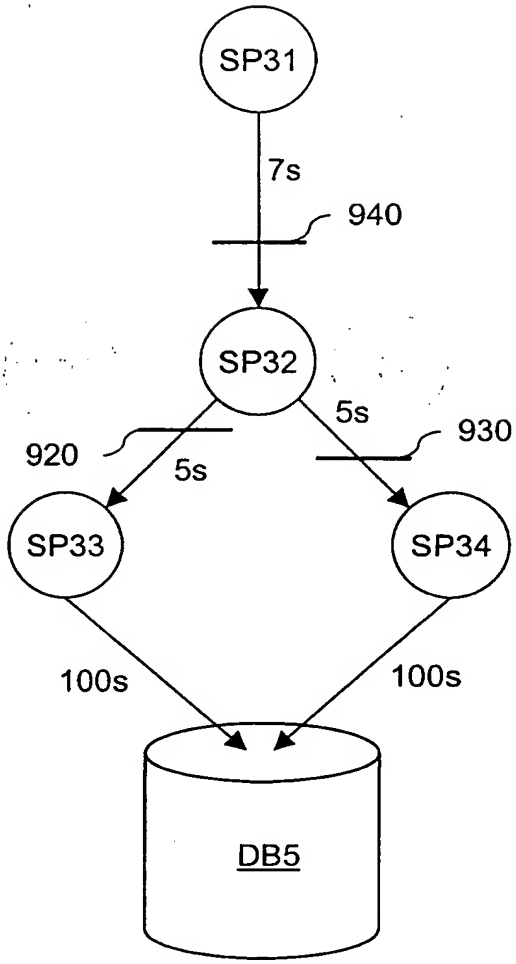

SP31
7s
940
SP32
5s
920
5s
930
SP33
SP34
100s
100s
DB5

FIG. 9b

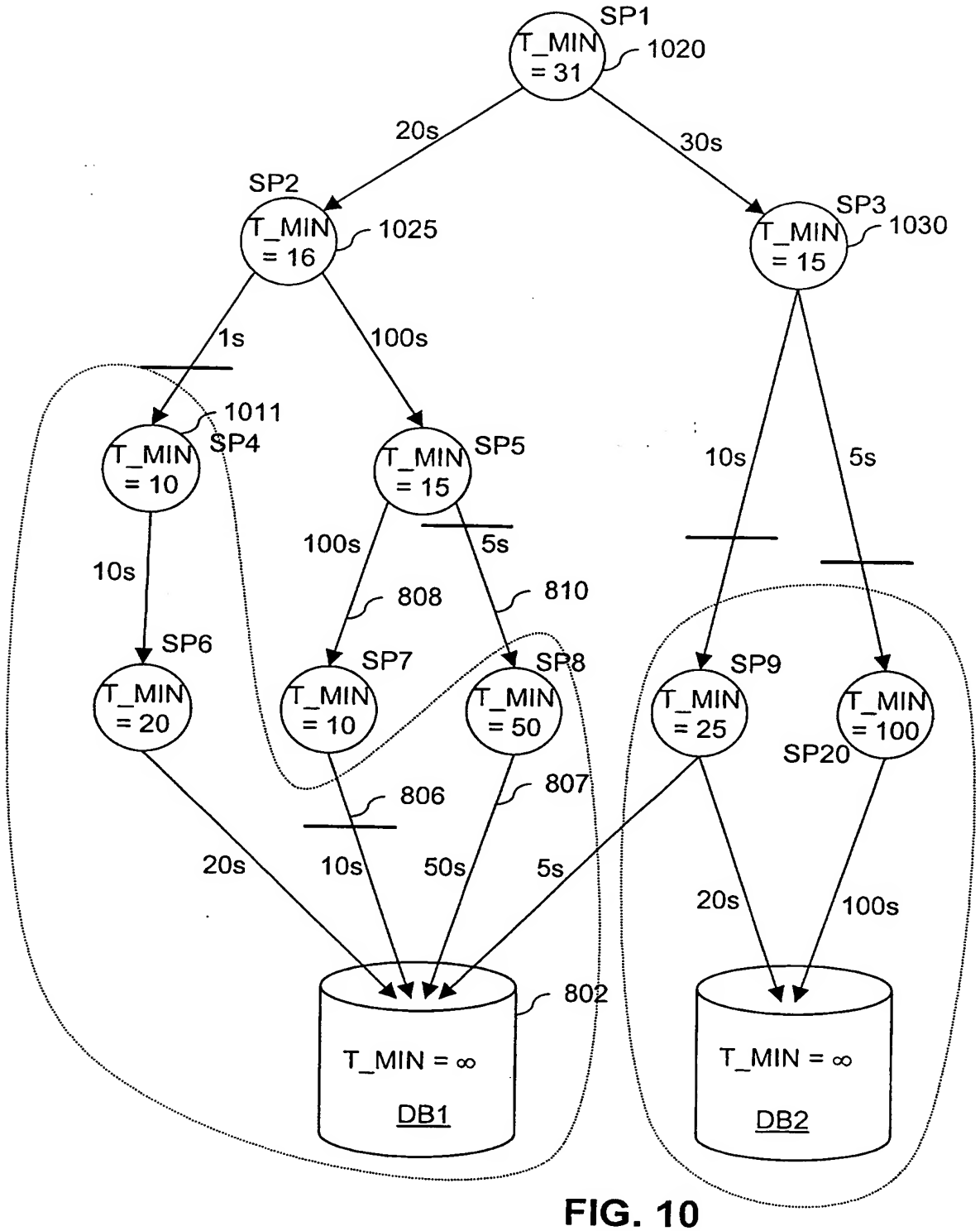

SP1
T_MIN = 31
1020
20s
30s
SP2
T_MIN = 16
1025
SP3
T_MIN = 15
1030
1s
100s
1011
SP4
T_MIN = 10
SP5
T_MIN = 15
10s
5s
100s
5s
10s
808
810
SP6
T_MIN = 20
SP7
T_MIN = 10
SP8
T_MIN = 50
SP9
T_MIN = 25
T_MIN = 100
SP20
806
807
20s
10s
50s
5s
20s
100s
802
T_MIN = ∞
DB1
T_MIN = ∞
DB2

FIG. 10

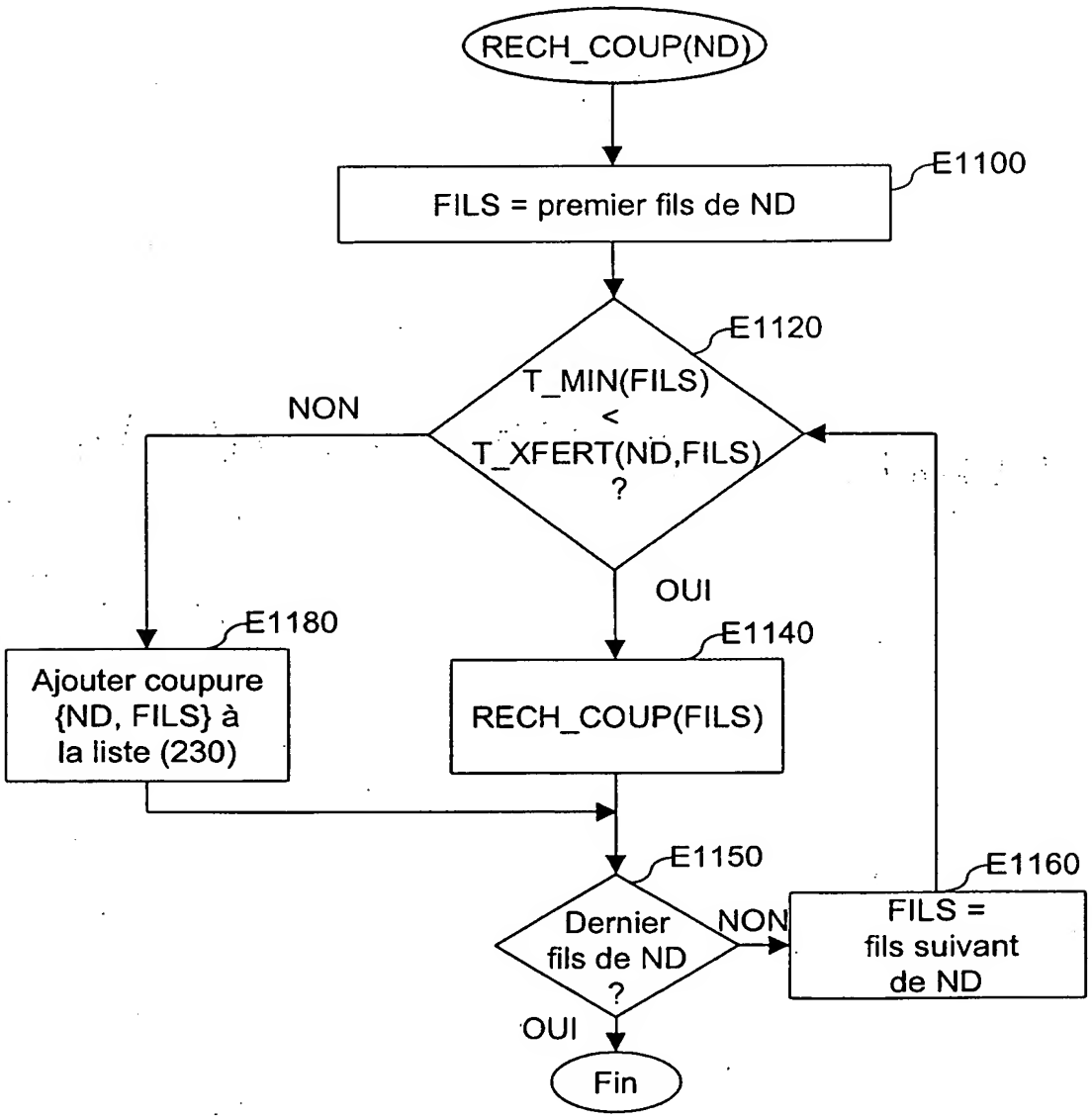

RECH_COUP(ND)
E1100
FILS = premier fils de ND
E1120
T_MIN(FILS)
<
T_XFERT(ND,FILS)
?
NON
OUI
E1180
Ajouter coupure
{ND, FILS} à
la liste (230)
E1140
RECH_COUP(FILS)
E1150
Dernier
fils de ND
?
NON
E1160
FILS =
fils suivant
de ND
OUI
Fin

FIG. 11

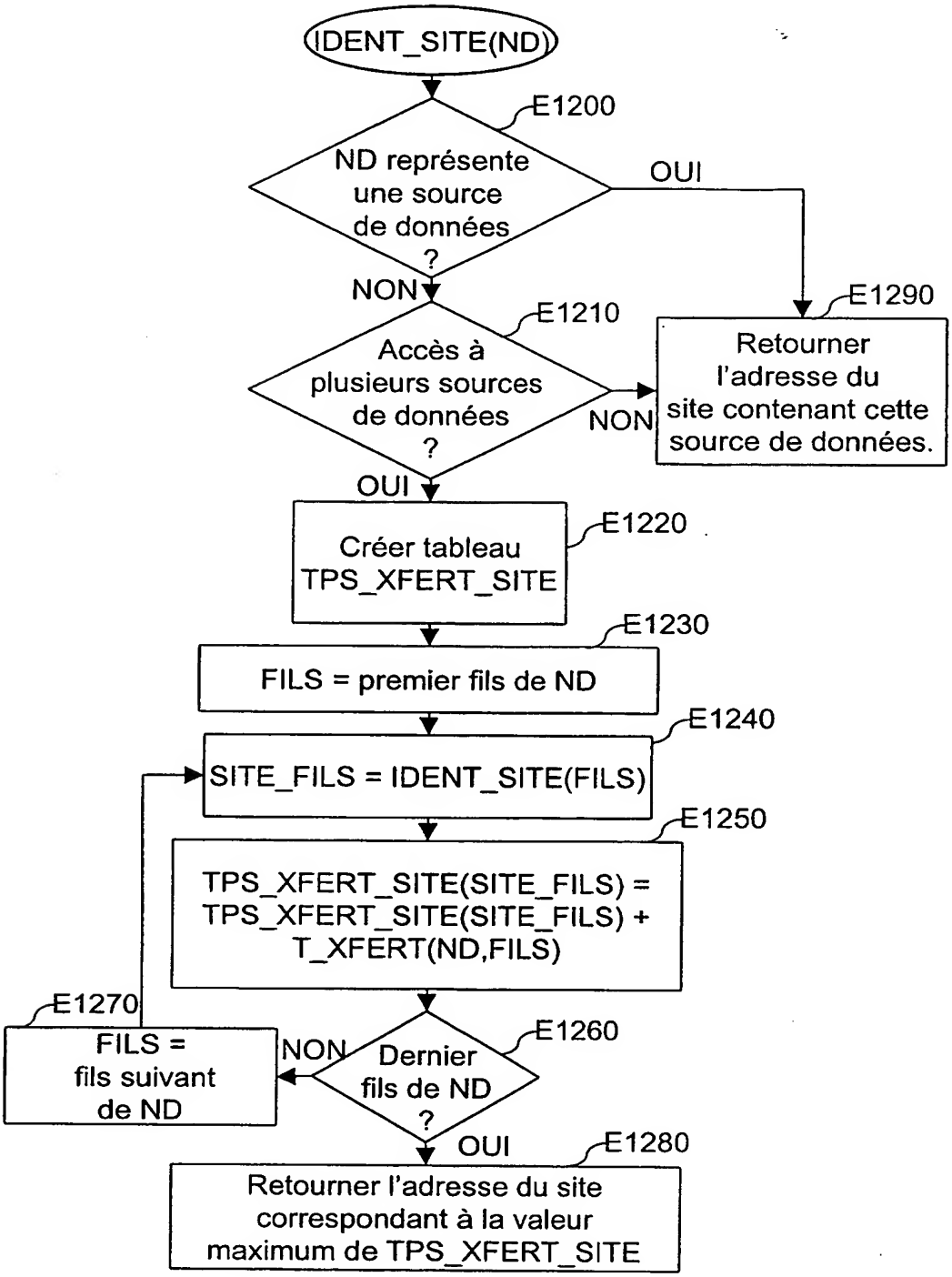

IDENT_SITE(ND)
E1200
ND représente une source de données ?
OUI
NON
E1210
Accès à plusieurs sources de données ?
NON
E1290
Retourner l'adresse du site contenant cette source de données.
OUI
Créer tableau TPS_XFERT_SITE
E1220
FILS = premier fils de ND
E1230
SITE_FILS = IDENT_SITE(FILS)
E1240
TPS_XFERT_SITE(SITE_FILS) = TPS_XFERT_SITE(SITE_FILS) + T_XFERT(ND,FILS)
E1250
E1270
FILS = fils suivant de ND
NON
Dernier fils de ND ?
E1260
OUI
E1280
Retourner l'adresse du site correspondant à la valeur maximum de TPS_XFERT_SITE

FIG. 12

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa N° 11235*02

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

DÉSIGNATION D'INVENTEUR(S) Page N° 1 / 1.
(Si le demandeur n'est pas l'inventeur ou l'unique inventeur)

Cet imprimé est à remplir lisiblement à l'encre noire

DB 113 W /260899

V s réf rences pour ce dossier *(facultatif)* 16 JAN 2001 75 INPI PARIS	BIF022759/FR
N° D'ENREGISTREMENT NATIONAL 0100532	01 00 532

TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Procédé et dispositif de partition de programme informatique.

LE(S) DEMANDEUR(S) :

CANON KABUSHIKI KAISHA

DESIGNE(NT) EN TANT QU'INVENTEUR(S) : (Indiquez en haut à droite «Page N° 1/1» S'il y a plus de trois inv nteurs, utilis z un formulaire identique et numérotez chaque page en indiquant le nombre total de pages).

Nom		RUELLAN
Prénoms		Hervé
Adresse	Rue	16, rue de la Chalotais
	Code postal et ville	35000 RENNES, France
Société d'appartenance *(facultatif)*		
Nom		MOREAU
Prénoms		Jean-Jacques
Adresse	Rue	91b, rue de Dinan
	Code postal et ville	35000 RENNES, France.
Société d'appartenance *(facultatif)*		
Nom		
Prénoms		
Adresse	Rue	
	Code postal et ville	
Société d'appartenance *(facultatif)*		

DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE (Nom t qualité du signataire)	Le 16 janvier 2001 Bruno QUANTIN N°92.1206 RINUY, SANTARELLI

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire.
Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.